

(11)Publication number : 2001-353678
(43)Date of publication of application : 25.12.2001

(21)Application number : 2000-175159	(71)Applicant : SONY CORP
(22)Date of filing : 12.06.2000	(72)Inventor : KASUGA TOMOAKI OKITA FUMIKO

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

【特許請求の範囲】

【請求項1】ロボットの動作制御プログラムを作成・編集するためのオーサリング・システムであって、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集部と、前記編集部を介して作成・編集されたシナリオを前記ロボット上で解釈可能なプログラム・コードに変換する変換部と、を具備することを特徴とするオーサリング・システム。

【請求項2】前記編集部は、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供することを特徴とする請求項1に記載のオーサリング・システム。

【請求項3】前記編集部は、所定のプログラム言語形式のスクリプトを作成・編集する環境をユーザに提供することを特徴とする請求項1に記載のオーサリング・システム。

【請求項4】前記変換部は、部品の配置により表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック・コードに変換することを特徴とする請求項1に記載のオーサリング・システム。

【請求項5】さらに、前記変換部において変換されたプログラム・コードを前記ロボットに転送するための通信手段を備えることを特徴とする請求項1に記載のオーサリング・システム。

【請求項6】前記通信手段は前記ロボットと無線データ通信を行うことを特徴とする請求項5に記載のオーサリング・システム。

【請求項7】前記通信手段は、前記変換部において変換されたプログラム・コードを1ステップずつ取り出して前記ロボットに転送することを特徴とする請求項5に記載のオーサリング・システム。

【請求項8】前記通信手段は、前記変換部において変換されたプログラム・コードを暗号化して前記ロボットに転送することを特徴とする請求項5に記載のオーサリング・システム。

【請求項9】さらに、前記通信手段を介して受信したプログラム・コードを解釈する解釈部と、前記解釈部による解釈結果に従って前記ロボットを駆動する駆動制御部と、を前記ロボット上に備えることを特徴とする請求項5に記載のオーサリング・システム。

【請求項10】前記解釈部は、受信したプログラム・コードを1ステップ単位で解釈して実行することを特徴とする請求項9に記載のオーサリング・システム。

【請求項11】前記ロボットは、現在の姿勢から直接遷移可能な動作や姿勢並びにある動作や姿勢を経由してな

ら遷移可能となる姿勢などに関する姿勢遷移制限情報を有し、

前記駆動制御部は、該姿勢遷移制限情報を基に、プログラム・コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項9に記載のオーサリング・システム。

【請求項12】前記姿勢遷移制限情報は、前記ロボットがとり得る姿勢を示すノードと、遷移可能な2つのノード間を結ぶ動作アークで構成される有向グラフ形式で保持され、

前記駆動制御部は、該有向グラフを探索してプログラム・コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項11に記載のオーサリング・システム。

【請求項13】ロボットの動作制御プログラムを作成・編集するためのオーサリング方法であって、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集ステップと、前記編集ステップを介して作成・編集されたシナリオを前記ロボット上で解釈可能なプログラム・コードに変換する変換ステップと、を具備することを特徴とするオーサリング方法。

【請求項14】前記編集ステップでは、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供することを特徴とする請求項13に記載のオーサリング方法。

【請求項15】前記編集ステップでは、所定のプログラム言語形式のスクリプトを作成・編集する環境をユーザに提供することを特徴とする請求項13に記載のオーサリング方法。

【請求項16】前記変換ステップでは、部品の配置により表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック・コードに変換することを特徴とする請求項13に記載のオーサリング方法。

【請求項17】さらに、前記変換ステップにおいて変換されたプログラム・コードを前記ロボットに転送するための通信ステップを備えることを特徴とする請求項13に記載のオーサリング方法。

【請求項18】前記通信ステップでは前記ロボットと無線データ通信を行うことを特徴とする請求項17に記載のオーサリング方法。

【請求項19】前記通信ステップは、前記変換ステップにおいて変換されたプログラム・コードを1ステップずつ取り出して前記ロボットに転送することを特徴とする請求項17に記載のオーサリング方法。

【請求項20】前記通信ステップでは、前記変換部において変換されたプログラム・コードを暗号化して前記ロ

ポットに転送することを特徴とする請求項17に記載のオーサリング方法。

【請求項21】さらに、前記通信ステップを介して受信したプログラム・コードを解釈する解釈ステップと、前記解釈ステップによる解釈結果に従って前記ロボットを駆動する駆動制御ステップと、を前記ロボット上で実行することを特徴とする請求項17に記載のオーサリング方法。

【請求項22】前記解釈ステップでは、受信したプログラム・コードを1ステップ単位で解釈して実行することを特徴とする請求項21に記載のオーサリング方法。

【請求項23】前記ロボットは、現在の姿勢から直接遷移可能な動作や姿勢並びにある動作や姿勢を経由してなら遷移可能となる姿勢などに関する姿勢遷移制限情報を有し、

前記駆動制御ステップでは、該姿勢遷移制限情報を基に、プログラム・コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項21に記載のオーサリング方法。

【請求項24】前記姿勢遷移制限情報は、前記ロボットがとり得る姿勢を示すノードと、遷移可能な2つのノード間を結ぶ動作アークで構成される有向グラフ形式で保持され、

前記駆動制御ステップでは、該有向グラフを探索してプログラム・コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項23に記載のオーサリング方法。

【請求項25】ロボットの動作制御プログラムを作成・編集するためのオーサリング処理をコンピュータ・システム上で実行するように記述されたコンピュータ・ソフトウェアをコンピュータ可読形式で物理的に格納した記憶媒体であって、前記コンピュータ・ソフトウェアは、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集ステップと、前記編集ステップを介して作成・編集されたシナリオを前記ロボット上で解釈可能なプログラム・コードに変換する変換ステップと、を具備することを特徴とする記憶媒体。

【請求項26】前記編集ステップでは、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供することを特徴とする請求項25に記載の記憶媒体。

【請求項27】前記編集ステップでは、所定のプログラム言語形式のスクリプトを作成・編集する環境をユーザに提供することを特徴とする請求項25に記載の記憶媒体。

【請求項28】前記変換ステップでは、部品の配置によ

り表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック・コードに変換することを特徴とする請求項25に記載の記憶媒体。

【請求項29】さらに、前記変換ステップにおいて変換されたプログラム・コードを前記ロボットに転送するための通信ステップを備えることを特徴とする請求項25に記載の記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、所定のシナリオに従ったデータを作成するためのオーサリング・システム及び方法に係り、特に、ロボットの所定の動作パターンを記述する一連のコマンド／データを作成するオーサリング・システム及び方法に関する。

【0002】更に詳しくは、本発明は、ロボットの動作状態を規定する部品の集合を用いて動作パターンを作成するオーサリング・システム及び方法に係り、特に、各部品をコンピュータ・ディスプレイ上に配置して動作パターンを作成するオーサリング・システム及び方法に関する。

【0003】

【従来の技術】電氣的若しくは磁氣的な作用を用いて人間の動作に似せた運動を行う機械装置のことを「ロボット」という。ロボットの語源は、スラブ語のROBOTA(奴隷機械)に由来すると言われている。わが国では、ロボットが普及し始めたのは1960年代末からであるが、その多くは、工場における生産作業の自動化・無人化などを目的としたマニピュレータや搬送ロボットなどの産業用ロボット(industrial robot)であった。

【0004】最近では、イヌやネコのように4足歩行の動物の身体メカニズムやその動作を模したペット型ロボット、あるいは、ヒトやサルなどの2足直立歩行を行う動物の身体メカニズムや動作を模した「人間形」若しくは「人間型」のロボット(humanoid robot)など、脚式移動ロボットやその安定歩行制御に関する研究開発が進展し、実用化への期待も高まってきている。これら脚式移動ロボットは、クローラ式ロボットに比し不安定で姿勢制御や歩行制御が難しくなるが、階段の昇降や障害物の乗り越え等、柔軟な歩行・走行動作を実現できるという点で優れている。

【0005】アーム式ロボットのように、ある特定の場所に植設して用いるような据置きタイプのロボットは、部品の組立・選別作業など固定的・局所的な作業空間でのみ活動する。これに対し、移動式のロボットは、作業空間は非限定的であり、所定の経路上または無経路上を自在に移動して、所定の若しくは任意の人的作業を代行したり、ヒトやイヌあるいはその他の生命体に置き換わる種々のサービスを提供することができる。

【0006】脚式移動ロボットの用途の1つとして、産

業活動・生産活動等における各種の難作業の代行が挙げられる。例えば、原子力発電プラントや火力発電プラント、石油化学プラントにおけるメンテナンス作業、製造工場における部品の搬送・組立作業、高層ビルにおける清掃、火災現場その他における救助といったような危険作業・難作業の代行などである。

【0007】また、脚式移動ロボットの他の用途として、上述の作業支援というよりも、生活密着型、すなわち人間との「共生」あるいは「エンターテインメント」という用途が挙げられる。この種のロボットは、ヒトあるいはイヌ（ペット）などの比較的知性の高い脚式歩行動物の動作メカニズムや四肢を利用した豊かな感情表現をエミュレートする。また、予め入力された動作パターンを単に忠実に実行するだけではなく、相手の言葉や態度（「褒める」とか「叱る」、「叩く」など）に呼応した、生き生きとした応答表現を実現することも要求される。

【0008】従来の玩具機械は、ユーザ操作と応答動作との関係が固定的であり、玩具の動作をユーザの好みに合わせて変更することはできない。この結果、ユーザは同じ動作しか繰り返さない玩具をやがては飽きてしまうことになる。

【0009】これに対し、知的なロボットは、動作に起因するモデルを備えており、外部からの音声や画像、触覚などの入力情報に基づいてモデルを変化させて動作を決定することにより、自律的な思考及び動作制御を実現する。ロボットが感情モデルや本能モデルを用意することにより、ロボット自身の感情や本能に従った自律的な行動を表出することができる。また、ロボットが画像入力装置や音声入出力装置を装備し、画像認識処理や音声認識処理を行うことにより、より高度な知的レベルで人間とのリアリスティックなコミュニケーションを実現することも可能となる。

【0010】また、ユーザ操作などの外部からの刺激を検出したことに応答してこのモデルを変更する、すなわち「学習効果」を付与することによって、ユーザにとって飽きない又は好みに適応した動作パターンを提供することができる。

【0011】昨今の脚式移動ロボットは高い情報処理能力を備えており、一種の計算機システムとして捉えることができる。したがって、ロボット上で実現される動作パターン、あるいは、複数の基本的な動作パターンの組合せによって構成される高度且つ複雑な一連の動作シーケンスは、コンピュータ・プログラミングと同様の作業によって構築される。

【0012】また、今後ますますロボットの普及率が高まり、産業界のみならず一般家庭や日常生活にも深く浸透していくことが予想される。とりわけ、エンターテインメント性を追求する製品に関しては、コンピュータやコンピュータ・プログラミングに関する高度な知識を持

たない一般消費者層がロボットを購入して使用するケースが多いと予想される。このような一般ユーザにとっても、ロボットの動作シーケンスを対話的な処理により比較的容易且つ効率的に作成・編集するためのことを支援するツール、すなわちオーサリング・システムを提供することが好ましいと考えられる。

【0013】

【発明が解決しようとする課題】本発明の目的は、ロボットの所定の動作パターンを記述する一連のコマンド／データを作成することができる、優れたオーサリング・システム及び方法を提供することにある。

【0014】本発明の更なる目的は、ロボットの動作状態を規定する部品の集合を用いて動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することにある。

【0015】本発明の更なる目的は、各部品をコンピュータ・ディスプレイ上に配置して動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することにある。

【0016】

【課題を解決するための手段】本発明は、上記課題を参照してなされたものであり、その第1の側面は、ロボットの動作制御プログラムを作成・編集するためのオーサリング・システム又は方法であって、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集部又は編集ステップと、前記編集部を介して作成・編集されたシナリオを前記ロボット上で解釈可能なプログラム・コードに変換する変換部又は変換ステップと、を具備することを特徴とするオーサリング・システム又は方法である。

【0017】前記編集部又は編集ステップは、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供するようにしてもよい。

【0018】あるいは、前記編集部又は編集ステップは、所定のプログラム言語形式のスク립トを作成・編集する環境をユーザに提供するようにしてもよい。

【0019】また、前記変換部又は変換部は、部品の配置により表現された動作シナリオ及び／又はスク립ト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック・コードに変換するようにしてもよい。ここで言うニーモニック・コードの一例はR CODEである。R CODEの詳細については後述に譲る。

【0020】また、前記変換部又は変換ステップにおいて変換されたプログラム・コードを前記ロボットに転送するための通信手段又は通信ステップをさらに備えてもよい。通信手段又は通信ステップでは、例えば、blue toothや、IIBなどのような近距離無線データ

通信を適用することができる。

【0021】ここで言う通信手段又は通信ステップは、前記変換部又は変換ステップにおいて変換されたプログラム・コードを1ステップずつ取り出して前記ロボットに転送するようにしてもよい。これに対し、ロボット側では、インタープリタを用いて受信したプログラム・コードを逐次解釈して実行及び／又はデバッグすることができる。

【0022】また、通信データのセキュリティを保つため、前記通信手段又は通信ステップは、前記変換部又は変換ステップにおいて変換されたプログラム・コードを暗号化して前記ロボットに転送するようにしてもよい。

【0023】また、前記通信手段又は通信ステップを介して受信したプログラム・コードを解釈する解釈部又は解釈ステップと、前記解釈部又は解釈ステップによる解釈結果に従って前記ロボットを駆動する駆動制御部又は駆動制御ステップとを前記ロボット上に備えてもよい。

【0024】このような場合、前記解釈部又は解釈ステップは、受信したプログラム・コードを1ステップ単位で解釈して実行するようにしてもよい。

【0025】脚式ロボットは一般に、現在の姿勢から直接遷移可能な動作や姿勢と、ある動作や姿勢を経由してなら遷移可能となる姿勢とを持つ。本発明に係るロボットは、現在の姿勢から直接遷移可能な動作や姿勢並びにある動作や姿勢を経由してなら遷移可能となる姿勢などに関する姿勢遷移制限情報をあらかじめ備えていてもよい。

【0026】このような場合、前記駆動制御部又は駆動制御ステップは、該姿勢遷移制限情報を基に、プログラム・コードの指示内容を姿勢遷移可能な形式に変換することができる。

【0027】また、かかる姿勢遷移制限情報は、前記ロボットがとり得る姿勢を示すノードと、遷移可能な2つのノード間を結ぶ動作アークで構成される有向グラフ形式で保持することができる。したがって、前記駆動制御部又は駆動制御ステップは、該有向グラフを探索することによって、プログラム・コードの指示内容を姿勢遷移可能な形式に容易に変換することができる。

【0028】また、本発明の第2の側面は、ロボットの動作制御プログラムを作成・編集するためのオーサリング処理をコンピュータ・システム上で実行するように記述されたコンピュータ・ソフトウェアをコンピュータ可読形式で物理的に格納した記憶媒体であって、前記コンピュータ・ソフトウェアは、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集ステップと、前記編集ステップを介して作成・編集されたシナリオを前記ロボット上で解釈可能なプログラム・コードに変換する変換ステップと、を具備することを特徴とする記憶媒体である。

【0029】本発明の第2の側面に係る記憶媒体は、例

えば、様々なプログラム・コードを実行可能な汎用コンピュータ・システムに対して、コンピュータ・ソフトウェアをコンピュータ可読な形式で物理的に提供する媒体である。このような媒体は、例えば、CD (Compact Disc) やFD (Floppy Disc)、MO (Magneto-Optical disc) などの着脱自在で可搬性の記憶媒体である。あるいは、ネットワーク（ネットワークは無線、有線の区別を問わない）などの伝送媒体などを経由してコンピュータ・ソフトウェアを特定のコンピュータ・システムにコンピュータ可読形式で提供することも技術的に可能である。

【0030】このような記憶媒体は、コンピュータ・システム上で所定のコンピュータ・ソフトウェアの機能を実現するための、コンピュータ・ソフトウェアと記憶媒体との構造上又は機能上の協働的關係を定義したものである。換言すれば、本発明の第2の側面に係る記憶媒体を介して所定のコンピュータ・ソフトウェアをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の第1の側面に係るオーサリング・システム又は方法と同様の作用効果を得ることができる。

【0031】

【作用】ユーザは、本発明を実現したオーサリング・ツールが提供するGUI画面を用いてマウス操作によりロボットの規定のシナリオを作成・編集する。あるいは、ユーザは、テキスト・エディタなどを用いて、スクリプト形式でロボットの動作制御プログラムを作成・編集してもよい。

【0032】オーサリング・ツールは、作成・編集したシナリオやスクリプト形式の動作制御プログラムをRCODEと呼ばれるニーモニック・コードに変換する。RCODE動作制御プログラムのデバッグ時には、RCODEプログラムを1行ごとに取り出して、暗号化して、blue toothや、11Bなどの無線通信手段を利用してロボット側に逐次転送する。

【0033】他方のロボット側では、RCODEなどで記述された動作制御プログラムの実行及びデバッグ環境として、インタープリタ／デバッガと、ミドルウェアと、ドライバなどを備えている。

【0034】インタープリタは、RCODE形式で記述されたプログラムを1行ずつ読み込んで解釈して実行する高水準言語プログラムである。デバッガは、RCODEプログラム中の誤り（バグ）を発見して、修正する作業を支援するプログラムである。ミドルウェアは、ロボットにおける歩行その他の脚式動作などの制御や、カメラからの入力画像の認識処理や、マイクからの音声入力の認識処理などを行う処理エンジンである。

【0035】ミドルウェアは、ロボット1が遷移可能な姿勢及び遷移する際の動作を、例えば有向グラフの形式であらかじめ登録しており、有向グラフに従って上位ブ

プログラムからの行動指令を姿勢遷移情報に変換して、ドライバに対する具体的な動作指示を発行するようになっている。

【0036】したがって、一般ユーザは、オーサリング・ツールを用いてGUI画面上でロボット1の動作シナリオを編集したり、あるいはスクリプト形式で動作制御プログラムを記述する限りにおいて、現実のロボットにおける姿勢・動作遷移特性のような詳細なハードウェア情報を気にしないで済む。また、ミドルウェア・レベルにおいてRCODEコマンド・レベルでの論理的な姿勢・動作指示とハードウェア上の姿勢・動作遷移特性との差異を吸収するので、ロボットの姿勢・動作遷移特性に反していることはRCODEプログラムのデバッグの対象とはならない。

【0037】本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【0038】

【発明の実施の形態】以下、図面を参照しながら本発明の実施例を詳解する。

【0039】A. オーサリング・システムの構成

図1には、本発明を実施に供される、四肢による脚式歩行を行う移動ロボット1の外観構成を示している。図示の通り、該ロボット1は、四肢を有する動物の形状や構造をモデルにして構成された多関節型の移動ロボットである。とりわけ本実施例の移動ロボット1は、愛玩動物の代表例であるイヌの形状及び構造を模してデザインされたベット型ロボットという側面を有し、例えば人間の住環境において人間と共存するとともに、ユーザ操作に応答した動作表現することができる。

【0040】移動ロボット1は、胴体部ユニット2と、頭部ユニット3と、尻尾4と、四肢すなわち脚部ユニット6A～6Dで構成される。

【0041】頭部ユニット3は、ロール、ピッチ及びヨーの各軸方向（図示）の自由度を持つ首関節7を介して、胴体部ユニット2の略前上端に配設されている。また、頭部ユニット3には、イヌの「目」に相当するCCD（Charge Coupled Device：電荷結合素子）カメラ15と、「耳」に相当するマイクロホン16と、「口」に相当するスピーカ17と、触感に相当するタッチセンサ18が搭載されている。これら以外にも、生体の五感を構成するセンサを含んでいても構わない。

【0042】尻尾4は、ロール及びピッチ軸の自由度を持つ尻尾関節8を介して、胴体部ユニット2の略後上端に湾曲若しくは揺動自在に取り付けられている。

【0043】脚部ユニット6A及び6Bは前足を構成し、脚部ユニット6C及び6Dは後足を構成する。各脚部ユニット6A～6Dは、それぞれ、大腿部ユニット9A～9D及び脛部ユニット10A～10Dの組み合わせで構成され、胴体部ユニット2底面の前後左右の各隅部

に取り付けられている。大腿部ユニット9A～9Dは、ロール、ピッチ、ヨーの各軸の自由度を持つ股関節11A～11Dによって、胴体部ユニット2の各々の所定部位に連結されている。また、大腿部ユニット9A～9Dと脛部ユニット10A～10Dの間は、ロール及びピッチ軸の自由度を持つ膝関節12A～12Dによって連結されている。

【0044】図示のように構成された移動ロボット1は、後述する制御部からの指令により各関節アクチュエータを駆動することによって、例えば、頭部ユニット3を上下左右に振らせたり、尻尾4を振らせたり、各足部ユニット6A～6Dを協調的に駆動させて歩行や走行などの動作を実現することができる。

【0045】なお、移動ロボット1の関節自由度は、実際には各軸毎に配備され関節アクチュエータ（図示しない）の回転駆動によって提供される。また、移動ロボット1が持つ関節自由度の個数は任意であり、本発明の要旨を限定するものではない。

【0046】図2には、移動ロボット1の電気・制御系統の構成図を模式的に示している。同図に示すように、移動ロボット1は、全体の動作の統括的制御やその他のデータ処理を行う制御部20と、入出力部40と、駆動部50と、電源部60とで構成される。以下、各部について説明する。

【0047】入出力部40は、入力部として移動ロボット1の目に相当するCCDカメラ15や、耳に相当するマイクロホン16、触感に相当するタッチセンサ18など、五感に相当する各種のセンサを含む。また、出力部として、口に相当するスピーカ17などを装備している。これら出力部は、脚などによる機械運動パターン以外の形式でロボット1からのユーザ・フィードバックを表現することができる。

【0048】移動ロボット1は、カメラ15を含むことで、作業空間上に存在する任意の物体の形状や色彩を認識することができる。また、移動ロボット1は、カメラのような視覚手段の他に、赤外線、音波、超音波、電波などの発信波を受信する受信装置をさらに備えていてもよい。この場合、各伝送波を検知するセンサ出力に基づいて発信源からの位置や向きを計測することができる。

【0049】駆動部50は、制御部20が指令する所定の運動パターンに従って移動ロボット1の機械運動を実現する機能ブロックであり、首関節7、尻尾関節8、股関節11A～11D、膝関節12A～12Dなどのそれぞれの関節におけるロール、ピッチ、ヨーなど各軸毎に設けられた駆動ユニットで構成される。図示の例では、移動ロボット1はn個の関節自由度を有し、したがって駆動部50はn個の駆動ユニットで構成される。各駆動ユニットは、所定軸回りの回転動作を行うモータ51と、モータ51の回転位置を検出するエンコーダ52と、エンコーダ52の出力に基づいてモータ51の回転

位置や回転速度を適応的に制御するドライバ53の組み合わせで構成される。

【0050】電源部60は、その字義通り、移動ロボット1内の各電気回路等に対して給電を行う機能モジュールである。本実施例に係る移動ロボット1は、バッテリーを用いた自律駆動式であり、電源部60は、充電バッテリー61と、充電バッテリー61の充放電状態を管理する充放電制御部62とで構成される。

【0051】充電バッテリー61は、例えば、複数本のニッケル・カドミウム電池セルをカートリッジ式にパッケージ化した「バッテリー・バック」の形態で構成される。

【0052】また、充放電制御部62は、バッテリー61の端子電圧や充電/放電電流量、バッテリー61の周囲温度などを測定することでバッテリー61の残存容量を把握し、充電の開始時期や終了時期などを決定するようになっている。充放電制御部62が決定する充電の開始及び終了時期は制御部20に通知され、移動ロボット1が充電オペレーションを開始及び終了するためのトリガとなる。充電オペレーションの詳細については後述に譲る。

【0053】制御部20は、「頭脳」に相当し、例えば移動ロボット1の頭部ユニット3あるいは胴体部ユニット2に搭載される。

【0054】図3には、制御部20の構成をさらに詳細に図解している。同図に示すように、制御部20は、メイン・コントローラとしてのCPU (Central Processing Unit) 21が、メモリその他の各回路コンポーネントや周辺機器とバス接続された構成となっている。バス27上の各装置にはそれぞれに固有のアドレス(メモリ・アドレス又はI/Oアドレス)が割り当てられており、CPU21はアドレス指定することでバス27上の特定の装置と通信することができる。

【0055】RAM (Random Access Memory) 22は、DRAM (Dynamic RAM) などの揮発性メモリで構成された書き込み可能メモリであり、CPU21が実行するプログラム・コードをロードしたり、作業データの一時的な保存のために使用される。

【0056】ROM (Read Only Memory) 23は、プログラムやデータを恒久的に格納する読み出し専用メモリである。ROM23に格納されるプログラム・コードには、移動ロボット1の電源投入時に実行する自己診断テスト・プログラムや、移動ロボット1の動作を規定する制御プログラムなどが挙げられる。

【0057】ロボット1の制御プログラムには、カメラ15やマイクロフォン16などのセンサ入力を処理する「センサ入力処理プログラム」、センサ入力と所定の動作モデルとに基づいて移動ロボット1の行動すなわち運動パターンを生成する「行動命令プログラム」、生成された運動パターンに従って各モータの駆動やスピーカ17の音声出力などを制御する「駆動制御プログラム」などが含まれる。生成される運動パターンには、通常の歩

行運動や走行運動以外に、「お手」、「お預け」、「お座り」や、「ワンワン」などの動物の鳴き声の発声などエンターテインメント性の高い動作を含んでいてもよい。

【0058】また、ロボット1のその他の制御プログラムとして、オーサリング・ツールを用いて作成・編集された各種の動作シーケンス・プログラムが含まれる。オーサリング・ツールは、例えば外部のコンピュータ・システム上で所定のソフトウェア実行環境下で起動するが、オーサリング・ツール並びに該ツール上で作成・編集されるプログラムについては後に詳解する。

【0059】不揮発性メモリ24は、例えばEEPROM (Electrically Erasable and Programmable ROM) のように、電氣的に消去再書き込みが可能なメモリ素子で構成され、逐次更新すべきデータを不揮発的に保持するために使用される。逐次更新すべきデータには、例えば、移動ロボット1の行動パターンを規定するモデルなどが挙げられる。

【0060】インターフェース25は、制御部20外の機器と相互接続し、データ交換を可能にするための装置である。インターフェース25は、例えば、カメラ15やマイクロフォン16、スピーカ17との間でデータ入出力を行う。また、インターフェース25は、駆動部50内の各ドライバ53-1...との間でデータやコマンドの入出力を行う。また、インターフェース25は、電源部60との間で充電開始及び充電終了信号の授受を行うこともできる。

【0061】インターフェース25は、RS (Recommended Standard) -232Cなどのシリアル・インターフェース、IEEE (Institute of Electrical and Electronics Engineers) 1284などのパラレル・インターフェース、USB (Universal Serial Bus) インターフェース、i-Link (IEEE1394) インターフェース、SCSI (Small Computer System Interface) インターフェースなどのような、コンピュータの周辺機器接続用の汎用インターフェースを備え、ローカル接続された外部機器との間でプログラムやデータの移動を行うようにしてもよい。

【0062】また、インターフェース25の1つとして赤外線通信 (IrDA) インターフェースを備え、外部機器と無線通信を行うようにしてもよい。赤外線通信のための送受信部は、例えば頭部ユニット2や尻尾3など、移動ロボット1本体の先端部に設置されることが受信感度の観点から好ましい。

【0063】さらに、制御部20は、無線通信インターフェース26ネットワーク・インターフェース・カード (NIC) 27を含み、「blue tooth」や、「11B」のような近接無線通信、あるいはLAN (Local Area Network: 例えばEthernet (登録商標)) やインターネットを経由して、外部のホスト・コンピュ

10

20

30

40

50

ータ100とデータ通信を行うことができる。

【0064】このようなロボットとホストコンピュータ間のデータ通信の目的は、遠隔のコンピュータ資源を用いて移動ロボット1の動作をリモート・コントロールすることである。また、該データ通信の他の目的は、動作モデルやその他のプログラム・コードなどロボット1の動作制御に必要なデータやプログラムをネットワーク経由で移動ロボット1に供給することにある。また、該データ通信の他の目的は、ホスト・コンピュータ100上でオーサリング・ツールを用いて作成・編集したロボット動作制御用プログラムのダウンロードや、このような動作制御用プログラムのホスト・コンピュータ100とロボット1との協働的動作によるリアルタイムのデバッグ処理である。但し、オーサリング・ツールやデバッグ処理については後に後述する。

【0065】また、制御部20は、テンキー及び／又はアルファベット・キーからなるキーボード29を備えておいてもよい。キーボード29は、作業現場での直接的なコマンド入力のために使用する他、パスワードなどの所有者認証情報の入力に用いられる。

【0066】本実施例に係るロボット1は、制御部20が所定の制御プログラムを実行することによって自律的な動作を行うことができる。また、画像入力（すなわちカメラ15）、音声入力（すなわちマイク16）、タッチセンサ18などの人間や動物の五感に相当する入力装置を備えるとともに、これら外部入力に応答した理性的又は感情的な動作を実行するインテリジェンスを備えている。

【0067】図1～図3に示すように構成された移動ロボット1は、以下のような特徴がある。すなわち、

【0068】（1）ある姿勢から他の姿勢へ遷移するように指示されたとき、各姿勢間を直接に遷移せず、あらかじめ用意された無理のない中間的な姿勢を経由して遷移する。

（2）姿勢遷移で任意の姿勢に到達したときに通知を受け取ることができる。

（3）頭部、足部、尻尾部などの各ユニット単位で姿勢を独立して管理し姿勢制御することができる。すなわち、ロボット1の全体の姿勢とは別に各ユニット毎に姿勢を管理することができる。

（4）動作命令の動作の詳細を示すためのパラメータを渡すことができる。

【0069】図3に示すように、本実施例に係る移動ロボット1は、ネットワーク経由で外部のホスト・コンピュータ100と相互接続されている。あるいは、ホスト・コンピュータ100とは、無線通信（例えば、bluetoothや、11B近距離無線データ通信）やその他の通信手段によって接続されていてもよい。

【0070】ホスト・コンピュータ100上では、所定のソフトウェア実行環境が構築され、該環境下では、オ

ーサリング・ツールを起動して、対話的な処理によりロボット1の動作シーケンスを比較的容易且つ効率的に作成することができる。オーサリング・ツールの詳細については後述する。

【0071】図4には、ホスト・コンピュータ100のハードウェア構成例を模式的に図解している。以下、コンピュータ100内の各部について説明する。

【0072】システム100のメイン・コントローラであるCPU（Central Processing Unit）101は、オペレーティング・システムOSの制御下で、各種のアプリケーションを実行するようになっている。OSは、より好ましくはGUI（Graphical User Interface）環境を提供するが、例えば、UNIX（登録商標）、又は、米Microsoft社のWindows 98/NTでよい。

【0073】図示の通り、CPU101は、バス107によって他の機器類（後述）と相互接続されている。バス107上の各機器にはそれぞれ固有のメモリ・アドレス又はI/Oアドレスが付与されており、CPU101はこれらアドレスによって機器アクセスが可能となっている。バス107の一例はPCI（Peripheral Component Interconnect）バスである。

【0074】メモリ102は、プロセッサ101において実行されるプログラム・コードを格納したり、実行中の作業データを一時保管するために使用される記憶装置である。同図に示すメモリ102は、不揮発及び揮発メモリ双方を含むものと理解されたい。

【0075】ディスプレイ・コントローラ103は、プロセッサ101が発行する描画命令を実際に処理するための専用コントローラであり、例えばSVG A（Super Video Graphic Array）又はXGA（eXtended Graphic Array）相当のビットマップ描画機能をサポートする。ディスプレイ・コントローラ103において処理された描画データは、例えばフレーム・バッファ（図示しない）に一旦書き込まれた後、表示装置111に画面出力される。表示装置111は、例えば、CRT（Cathode Ray Tube）ディスプレイや、液晶表示ディスプレイ（Liquid Crystal Display）などである。

【0076】入力機器インターフェース104は、キーボード112やマウス113などのユーザ入力機器をシステム100に接続するための装置である。入力機器インターフェース104は、キーボード112によるキー入力又はマウス113を介した座標指示入力にตอบสนองして、プロセッサ101に対して割り込みを発生する。

【0077】ネットワーク・インターフェース105は、Ethernetなどの所定の通信プロトコルに従って、システム100をLAN（Local Area Network）などのネットワークに接続したり、あるいはbluetoothや、11Bのような近距離無線データ通信に接続することができる。ネットワーク・インターフェース

105は、一般に、LANアダプタ・カードの形態で提供され、マザーボード（図示しない）上のPCIバス・スロットの装着して用いられる。

【0078】図3に示す例では、ホスト・コンピュータ100は、無線データ通信やネットワーク経由でロボット1と相互接続されているが、勿論、他の通信手段によって両者が接続されていてもよい。

【0079】またネットワーク上では、複数のホスト・コンピュータ（図示しない）がトランスペアレントな状態で接続され、分散コンピューティング環境が構築されている。ネットワーク上では、ソフトウェア・プログラムやデータ・コンテンツなどの配信が行われる。例えば、本実施例に係るオーサリング・ツールや、該オーサリング・ツールで作成・編集したロボット用動作シーケンス・プログラムなどを、ネットワーク経由で配信することができる。また、このようなプログラム／データの配信サービスを有料又は無料で行ってもよい。

【0080】外部機器インターフェース106は、ハード・ディスク・ドライブ（HDD）114やメディア・ドライブ115などの外部装置をシステム100に接続するための装置である。外部機器インターフェース106は、例えば、IDE（Integrated Drive Electronics）やSCSI（Small Computer System Interface）などのインターフェース規格に準拠する。

【0081】HDD114は、記憶担体としての磁気ディスクを固定的に搭載した外部記憶装置であり（周知）、記憶容量やデータ転送速度などの点で他の外部記憶装置よりも優れている。ソフトウェア・プログラムを実行可能な状態でHDD114上に置くことをプログラムのシステムへの「インストール」と呼ぶ。通常、HDD114には、プロセッサ511が実行すべきオペレーティング・システムのプログラム・コードや、アプリケーション・プログラム、デバイス・ドライバなどが不揮発的に格納されている。例えば、本実施例に係るオーサリング・ツールや、該オーサリング・ツールを用いて作成・編集したロボット用動作シーケンス・プログラムを、HDD114上にインストールすることができる。

【0082】また、メディア・ドライブ115は、CD（Compact Disc）やMO（Magneto-Optical disc）、DVD（Digital Versatile Disc）などの可搬型メディアを装填して、データ記録面にアクセスするための装置である。可搬型メディアは、主として、ソフトウェア・プログラムやデータ・ファイルなどをコンピュータ可読形式のデータとしてバックアップすることや、これらをシステム間で移動（販売・流通・配布を含む）する目的で利用される。例えば、本実施例に係るオーサリング・ツールや、該オーサリング・ツールを用いて作成したロボット用動作シーケンス・プログラムなどを、これら可搬型メディアを利用して機器間で物理的に流通・配布することができる。

【0083】なお、図4に示すようなホスト・コンピュータ100の一例は、米IBM社のパーソナル・コンピュータPC/AT（Personal Computer/Advanced Technology）の互換機又は後継機である。勿論、他のアーキテクチャを備えた計算機システムを本実施例に係るホスト・コンピュータ100に適用することも可能である。

【0084】本実施例では、ロボット1の所定の動作パターンを記述する一連のコマンド／データからなる動作制御プログラムを、ホスト・コンピュータ100上で起動したオーサリング・ツールを用いて作成・編集する。また、このオーサリング・ツールを用いて作成・編集した動作制御プログラムを、例えばbluetoothや、11Bなどの無線通信手段を用いてロボット1側に転送して、ホスト・コンピュータ100とロボット1との協働的動作によりデバッグ処理を行うようになっている。すなわち、ホスト・コンピュータ100とロボット1の誘起的な結合により、ロボット1の動作制御プログラムのためのオーサリング・システムが構築される。

【0085】図5には、オーサリング・システムの全体構成を模式的に図解している。

【0086】ホスト・コンピュータ100側では、ユーザは、オーサリング・ツールが提供するGUI（Graphical User Interface）画面を用いてマウス操作によりロボット1の規定のシナリオを作成・編集することができる。シナリオ作成用のGUI画面並びに該画面上の操作については後述する。あるいは、ユーザは、テキスト・エディタなどを用いて、スクリプト形式（例えばC言語などの高級言語形式）でロボット1の動作制御プログラムを作成・編集することができる。

【0087】オーサリング・ツールは、ユーザがGUI画面上で作成・編集したシナリオや、テキスト・エディタ上で作成・編集したスクリプト形式の動作制御プログラムを、“RCODE”と呼ばれるアセンブラのようなニーモニック・コードに変換する。

【0088】RCODEとは、簡単なコマンドでロボット1を制御するために策定されたプログラム言語であり、IFやGOなどの基本的な制御構造も備えているので、ロボット制御用最低水準スクリプト言語としても使用することができる。但し、RCODEの詳細については後に詳解する。

【0089】ホスト・コンピュータ100上で作成・編集されたRCODE動作制御プログラムは、例えば、メモリ・スティックなどのメディアを利用してロボット1側に移動することができる。また、RCODE動作制御プログラムのデバッグ時には、RCODEプログラムを1行ごとに取り出して、暗号化して、bluetoothや、11Bなどの無線通信手段を利用してロボット1側に逐次転送するようになっている。

【0090】他方、ロボット1側では、RCODEなどで記述された動作制御プログラムの実行及びデバッグ環

境として、インタープリタ／デバッガと、ミドルウェアと、ドライバと、オペレーティング・システム（OS）とを備えている。

【0091】インタープリタは、RCODE形式で記述されたプログラムを1行ずつ読み込んで解釈して実行する高水準言語プログラムである。但し、デバッグ時などにおいて、ホスト・コンピュータ100側から暗号化された形式でRCODEプログラムが送信される場合には、インタープリタは、これを一旦復号化してから解釈・実行を行う。

【0092】デバッガは、RCODEプログラム中の誤り（バグ）を発見して、修正する作業を支援するプログラムである。すなわち、プログラムを指定した行で実行を止めたり、そのときのメモリや変数の内容を参照することができる。

【0093】ミドルウェアは、ロボット1における歩行その他の脚式動作などの制御や、カメラ15からの入力画像の認識処理や、マイク16からの音声入力の認識処理などを行う処理エンジンである。

【0094】ドライバは、各関節アクチュエータやその他のハードウェアの操作を行うためのプログラム・コードである。

【0095】本実施例では、ミドルウェア並びにドライバは、オブジェクト指向プログラムによって実装されている。オブジェクト指向に基づくソフトウェアは、基本的に、データとそのデータに対する処理手続きとを一体化させた「オブジェクト」というモジュール単位で扱われる。また、必要に応じて複数のオブジェクトを作成したり組み合わせることで1つのソフトウェアが完成する。一般に、オブジェクト指向によれば、ソフトウェアの開発と保守が効率化されると考えられている。

【0096】オペレーティング・システム（OS）は、これらオブジェクト間のデータ通信の管理や、その他のプログラム実行に関する制御を行う。OSもオブジェクト指向プログラムにより実装される。

【0097】B. オーサリング・ツールを用いたロボット用動作プログラムの作成・編集

次いで、本実施例に係るオーサリング・ツールが提供する、ユーザがロボット1の動作シナリオを作成・編集するためのGUI画面について説明する。図6には、該GUI画面の構成を図解している。

【0098】同図に示すように、該GUI画面は、マウスによるクリック、ドラッグ&ドロップ操作を基調として編集作業を行う編集ウィンドウと、編集作業において頻繁に使用される動作手順を部品化して提供する部品ウィンドウを含んでいる。

【0099】編集ウィンドウは、最上段より順に、タイトル・バー、メニュー・バー、ツール・バー、編集領域が配設されている。

【0100】メニュー・バーは、ユーザがクリック操作

により選択可能な処理の一覧を横方向に配列した領域であり、この例では、「ファイル（F）」、「編集（E）」、「表示（V）」、「挿入（I）」、「グループ（G）」という各メニュー項目が用意されている。あるメニュー項目を選択すると、さらに該当するプルダウン・メニューが出現する。

【0101】図7には、メニュー項目「ファイル」に関するプルダウン・メニューを図解している。同図に示すように、該プルダウン・メニュー中には、「新規作成」、「開く」、「上書き保存」、「名前を付けて保存」、「メモリスティックイメージの作成」、「メモリスティックイメージの転送」、「終了」という各サブメニューが配列されている。

【0102】「新規作成」を選択すると、ロボットの一連の動作を記述するシナリオ（以下、「プロジェクト」とも言う）を新規に作成する。既に編集中のプロジェクトが編集ウィンドウ上に存在する場合には、その保存を促すダイアログが出現する。図6に示す編集ウィンドウは新規プロジェクトを作成したときの初期画面であり、プロジェクトの先頭と最後尾にそれぞれ該当する“START”ボックス及び“END”ボックスのみからなるメイン・グループが編集領域に表示されている。

【0103】サブメニュー「開く」を選択すると、ファイルとして保存されたプロジェクトを編集ウィンドウ上に開く。既に編集中のプロジェクトが編集ウィンドウ上に存在する場合には、その保存を促すダイアログが表示される。本実施例では、プロジェクトは、拡張子“apk”を持つファイルとして扱われるものとする。

【0104】サブメニュー「上書き保存」を選択すると、同名のファイルに編集中のプロジェクトを保存する。このとき、ファイル名を尋ねるダイアログは出現しないものとする。

【0105】サブメニュー「名前を付けて保存」を選択すると、ファイル名を尋ねるダイアログが表示される。該ダイアログ上で、ユーザは保存先のディレクトリ名とファイル名を指定することができ、指定されたディレクトリにそのファイル名でプロジェクトが保存される。

【0106】サブメニュー「メモリスティックイメージの作成」を選択すると、メモリ・スティック（又はこれと等価な記憶メディア）に書き込むためのイメージ・ファイルを作成する。本実施例では、イメージ・ファイルは、元のapk形式のファイルと同じディレクトリに“aim”という拡張子で保存されるものとする。

【0107】サブメニュー「メモリスティックイメージの転送」を選択すると、メモリ・スティック上のイメージ・ファイルを指定されたフルバスに生成する。

【0108】サブメニュー「終了」を選択すると、実行中のオーサリング・ツールを終了する。

【0109】また、図8には、メニュー項目「グループ」に関するプルダウン・メニューを図解している。同

図に示すように、該プルダウン・メニュー中には、「新規グループ作成」、「このグループの削除」、「グループ一覧」、「グループ階層構造」、「全てのグループを開く」、「全てのグループを閉じる」という各サブメニューが配列されている。

【0110】サブメニュー「新規グループ作成」を選択すると、新規グループの作成を行う。

【0111】サブメニュー「このグループの削除」を選択すると、現在表示されているグループの削除を行う。但し、グループを削除しても、そのグループを呼び出し 10 ているグループ・ボックスは削除されない。

【0112】サブメニュー「グループ一覧」を選択すると、グループの一覧を表示する。表示されたグループの名前をダブルクリックすることで、そのグループを開くことができる。

【0113】サブメニュー「グループ階層構造」を選択すると、グループの階層構造（呼び出し構造）を表示する。表示されたグループの名前をダブルクリックすることで、そのグループを開くことができる。

【0114】サブメニュー「全てのグループを開く」を 20 選択すると、すべてのグループのウィンドウを開く。

【0115】サブメニュー「全てのグループを閉じる」を選択すると、すべてのグループのウィンドウを閉じる（但し、メイン・グループは除く）。

【0116】また、図6に示す例では、ツール・バー内には、「通常モード」、「接続モード」、「動作ボックス」、「分岐ボックス」、「グループ・ボックス」という各ツール・ボタンが配設されている。

【0117】ボタン「通常モード」を選択すると、マウス操作が通常モードに遷移する。例えば、ボックスに対 30 するドラッグ動作は、ボックスの移動として機能する。

【0118】ボタン「接続モード」を選択すると、マウス操作が接続モードに遷移する。すなわち、ボックスに対するドラッグ動作は、ボックス間の接続として機能する。接続元から接続先のボックスに対してドラッグ動作を行えばよい。また、分岐ボックスからの接続である場合、分岐条件を指定するダイアログが表示される。

【0119】ボタン「動作ボックス」を選択すると、動作 (Action) ボックスが下方の編集領域に挿入される。

【0120】ボタン「分岐ボックス」を選択すると、分岐 (Branch) ボックスが下方の編集領域に挿入される。

【0121】ボタン「グループ・ボックス」を選択すると、グループ・ボックスが下方の編集領域に挿入される。

【0122】ユーザは、図6に示す編集領域上で、クリック、ドラッグなどのマウス操作や、マウス操作とキー入力の組合せにより、処理を指定することができる。

【0123】例えば、マウスの左ボタンのクリックによ 50

りボックスを選択することができる。

【0124】また、マウスの左ボタンをダブル・クリックしたときは、プロパティを表示し、及び／又は、グループを開く。

【0125】また、通常モード下でマウスの左ボタンを押下してドラッグしたとき、マウス・カーソルがボックス上にあったときには、ボックスの移動を行い、マウス・カーソルがボックス外にあったときには範囲選択を行う。また、接続モード下でマウスの左ボタンを押下してドラッグしたときには、マウス・カーソルがボックス上 10 にあるときにはボックスどうしの接続を行う。

【0126】また、通常モード下で [Ctrl] キー押下とマウスの左ボタンの押下／ドラッグの組合せを行うと、マウス・カーソルがボックス上にあったときにはボックスの接続を行い、マウス・カーソルがボックス外にあったときには範囲指定を行う。また、接続モード下で [Ctrl] キー押下とマウスの左ボタンの押下／ドラッグの組合せを行うと、マウス・カーソルがボックス上 20 にあるときにはボックスの移動を行う。

【0127】また、[Shift] キー押下とマウスの左ボタン押下／ドラッグの組合せを行うと、編集ウィンドウのリサイズを行う。

【0128】また、マウスの右ボタンをクリックすると、コンテキスト・メニューが表示される。

【0129】なお、マウスのドラッグ操作は、[Esc] キーの押下でキャンセルされる。

【0130】次いで、図6に示すような、オーサリング・ツールが提供するGUI画面上における基本操作手順について説明する。

【0131】まず、メニュー[ファイル]のサブメニュー[新規作成]を選択することにより、新規プロジェクトを作成する。新規プロジェクトの画面は、図6に示すように、プロジェクトの先頭と最後尾にそれぞれ該当する"START"ボックス及び"END"ボックスのみからなるメイン・グループが編集領域に表示されている。

【0132】次いで、メニュー[挿入]又はツール・バー内のボタン[A]、[B]、[G]を選択することにより、編集ウィンドウ上に該当するボックスを配置することができる。一旦編集ウィンドウ上に配置されたボックスは、マウス操作により設置位置を移動することがで 30 きる。

【0133】図9には、編集ウィンドウ上に動作ボックス[A]及び分岐ボックス[B]が各1個配置されている様子を図解している。各ボックスには、デフォルトの名前"Box001"並びに"Box002"が付されているが、これはユーザが後で変更することができる。

【0134】次いで、メニュー[編集]のサブメニュー[マウス：接続モード]を選択するか、又は、ツール・バー内の[接続モード]ボタンを選択することにより、マウス操作を接続モードに切り換える。そして、マウス 50

操作により、シナリオ進行上の前後関係にあるボックスどうしを接続する。

【0135】図10には、図9に示す編集ウィンドウ上で、[START]ボックスと動作ボックス[A]、並びに、動作ボックス[A]と分岐ボックス[B]を接続した様子を図解している。

【0136】ボックスどうしを接続する際、分岐ボックス[B]からの接続に関しては、分岐上辺を指定するためのダイアログ（図示しない）が自動的に開くようになっているので、ユーザは、ダイアログ内の指示に従って条件を指定していけばよい。但し、分岐ボックスの条件指定は、次のボックスへの接続時よりも後回しにはできないものとし、また、条件を完全に指定しないと接続が行われない。

【0137】また、ユーザは、あるボックス上でマウスの左ボタンをダブルクリックすることで、当該ボックスの処理動作の詳細を指定するためのダイアログが開くようになっている。また、グループ・ボックスをダブルクリックした場合には、そのグループが別の編集ウィンドウとして開くようになっている（後述）。

【0138】本実施例では、各ボックスの処理動作は、RCODEと呼ばれる、アセンブラのようなニーモニック・コードによって記述される。RCODEとは、簡単なコマンドでロボット1を制御するために策定されたプログラム言語であるが、RCODEの詳細については後に詳解する。

【0139】図11には、動作ボックスの詳細を指定するためのダイアログを図解している。このダイアログは、例えば図9又は図10に示すような編集ウィンドウ上で、所望の動作ボックス[A]をダブルクリックすることによって呼び出すことができる。

【0140】ユーザは、図11に示すようなダイアログ上で、名前フィールドに文字列を入力することによって動作ボックスの名前を指定することができる。また、コメント・フィールド上に、該動作ボックスに関するコメントを記入することができる。

【0141】さらに、Action、Part、Sound、Volume#などのコンボボックス内に、RCODEのコマンド名やコマンド・パラメータを直接入力するか、又は、該ボックスの右端の▼ボタンを押して出現するリスト・ボックス（図示しない）から所望のコマンド又はコマンド・パラメータを選択することによって、1行すなわち1ステップ分のRCODEコマンドを設定することができる。

【0142】これらAction、Part、Sound、Volume#などのコンボボックスを利用して1ステップ分のRCODEコマンドを設定し、さらに[追加(A)]ボタンをクリックすると、コマンド・リスト上に順次登録される。

【0143】また、このコマンド・リスト上で所定の行

を選択してから、[変更(M)]ボタンをクリックすることによって、当該行は変更対象となり、その設定内容がAction、Part、Sound、Volume#などの各コンボボックスに表示される。また、このコマンド・リスト上で所定の行を選択してから、[削除(D)]ボタンをクリックすることによって、当該行をコマンド・リストから削除することができる。

【0144】そして、このダイアログ上で動作ボックスの詳細の指定を完了すると、[閉じる(C)]ボタンをクリックすることによって、画面を介した指定内容がRCODEで記述された処理ルーチンとして登録されるとともに、ダイアログは閉じる。

【0145】また、図12には、分岐ボックスの詳細を指定するためのダイアログを図解している。このダイアログは、例えば図9又は図10に示すような編集ウィンドウ上で、所望の分岐ボックス[B]をダブルクリックすることによって呼び出すことができる。

【0146】ユーザは、図12に示すようなダイアログ上で、名前フィールドに文字列を入力することによって分岐ボックスの名前を指定することができる。図示の例では、当該分岐ボックスの名前として「モード分岐」が記入されている。また、コメント・フィールド上に、該分岐ボックスに関するコメントを記入することができる。

【0147】さらに、Type、Variableなどのコンボボックス内に、RCODEのコマンド名やコマンド・パラメータを直接入力するか、又は、該ボックスの右端の▼ボタンを押して出現するリスト・ボックス（図示しない）から所望のコマンド又はコマンド・パラメータを選択することによって、当該分岐ボックスの条件判断を記述するRCODEコマンドを設定することができる。

【0148】これらType、Variableなどのコンボボックスを利用して分岐条件のRCODEコマンドを設定し、さらに[追加(A)]ボタンをクリックすると、コマンド・リスト上に順次登録される。

【0149】また、このコマンド・リスト上で所定の行を選択してから、[変更(M)]ボタンをクリックすることによって、当該行は変更対象となり、その設定内容がType、Variableなどの各コンボボックスに表示される。また、このコマンド・リスト上で所定の行を選択してから、[削除(D)]ボタンをクリックすることによって、当該行をコマンド・リストから削除することができる。

【0150】そして、このダイアログ上で分岐ボックスの詳細の指定を完了すると、[閉じる(C)]ボタンをクリックすることによって、指定内容がRCODEで記述された処理ルーチンとして登録されるとともに、ダイアログは閉じる。

【0151】なお、ユーザは、図11や図12に示すよ

うなダイアログを使用して各ボックスの詳細を自ら設定する必要は必ずしもない。オーサリング・ツールが用意するGUI画面上の部品ウィンドウ（例えば図6を参照のこと）には、あらかじめ詳細な指定を完了した動作ボックスや分岐ボックスなどが部品化して用意されているので、ユーザは、所望の部品ボックスを部品ウィンドウから編集ウィンドウにドラッグ・アンド・ドロップすることにより、そのまま当該部品をプログラムの一部としてそのまま使用することができる。

【0152】また、オーサリング・ツールが提供するGUI画面（図6を参照のこと）で、あるボックス上でマウスの右ボタンをクリックすると「コンテキスト・メニュー」（図示しない）が出現する。このコンテキスト・メニューからは、[ボックスの削除]や[接続の削除]を行うことができる。

【0153】但し、GUI画面上で、ボックス間を接続する特定の線を指定して、その削除を直接指示することはできず、線の削除には必ずコンテキスト・メニューを使用しなければならない。分岐ボックスの場合、どの接続を削除するのかわを選択するためのリスト・ボックスを含んだダイアログ（図示しない）が出現するようになっている。ユーザは、該リストの中から削除したい接続を選んだ[削除]ボタンをクリックすればよい。

【0154】上述したようなオーサリング・ツールのGUI画面を用いて、ユーザは、例えばロボット1がサッカー・ゲームを行うようなシナリオすなわち動作手順を記述したプログラムを作成することができる。このようなシナリオは、例えば、図13に示すようなボックス構成で記述される。

【0155】図13に示す例では、動作ボックス[初期設定]において指定された内容でロボット1の初期設定を行った後、分岐ボックス[モード分岐]で指定された*

《制御》

EDIT 編集モードに移行する（シリアル・ラインを通してのコードの転送）。

RUN 実行を開始する。

END 編集モードを終了する／実行を終了する。

LOAD:<filename> メモリ・スティック上のコードをメモリ上に読み出す

SAVE:<filename> メモリ上のコードをメモリ・スティック上に書き込む

《ジャンプ》

:<label> ラベル定義。<label>は1以上の数値（0は次の行を表す）。

GO:<label> ラベルにジャンプする。

《分岐》

IF:<op>:<var1>:<const>|<var2>:<then_label>[:<else_label>]

変数1<var1>と定数<const>若しくは変数2<var2>を比較し、分岐する。

<op> は、"=", "<", ">", "<=", ">=", ">"の6種類がある。

SWITCH:<var>

変数の値をコンテキスト値として設定する。

CSET:<op>:<var1>|<const1>:<var2>|<const2>:<var3>|<const3>

値1と値2を<op>演算子で比較して、その結果が真ならば、値3をコンテキスト値として設定する。

*条件との比較結果に応じて、ロボット1は[転倒回復]、[探索モード]、[追跡モード]のうち1つの動作モードを実行するようになっている。

【0156】ここで、[転倒回復]、[探索モード]、[追跡モード]の各々はグループ・ボックスとして記述されている。前述したように、編集ウィンドウ上でグループ・ボックスをダブルクリックした場合には、そのグループが別の編集ウィンドウとして開くようになっている。図14には、このうち[探索モード]をダブルクリックして出現する、探索モードの編集ウィンドウを図解している。また、図15には、[追跡モード]をダブルクリックして出現する、追跡モードの編集ウィンドウを図解している。

【0157】本実施例では、ロボット1の処理動作は、RCODEと呼ばれる、アセンブラのようなニーモニック・コードによって記述されるということは既に述べた。ここで、RCODEの詳細について説明しておく。

【0158】RCODEとは、簡単なコマンドでロボット1を制御するために策定されたプログラム言語であり、IFやGOなどの基本的な制御構造も備えているので、ロボット制御用最低水準スクリプト言語としても使用することができる。

【0159】RCODEでは、すべての大文字の単語は定数名などのために予約されており、ユーザが用意する変数名などに使用することができない。

【0160】RCODEは、制御、ジャンプ、分岐、代入、同期、関数など、実行すべき行動を指定するための演算子(operator)を備えている。以下に、RCODE演算子の一覧を示しておく。

【0161】

[数1]

CSET命令が連続して現れた場合、真になったCSET命令以降のCSET命令はNo Operation扱いとなる（縦積み命令）。

CASE:<const>:<RCODE command>

コンテキスト値が<const>と等しければ<RCODE command>を実行する。

《代入》

LET:<var1>:<const>|<var2> <var1> ← <const>|<var2>

変数への値の代入

SET:<var1>:<const>|<var2> <var1> ← <const>|<var2>

変数によっては特殊機能が働く。（単なる代入ではない）

GET:<var1>

<var1>の内容をコンソールに表示（デバッグ用）

《演算》

ADD:<var1>:<const>|<var2> <var1>←<var1>+<const>|<var2> 加算

SUB:<var1>:<const>|<var2> <var1>←<var1>-<const>|<var2> 減算

MUL:<var1>:<const>|<var2> <var1>←<var1>*<const>|<var2> 乗算

DIV:<var1>:<const>|<var2> <var1>←<var1>/<const>|<var2> 除算

MOD:<var1>:<const>|<var2> <var1>←<var1>%<const>|<var2> 剰余

AND:<var1>:<const>|<var2> <var1>←<var1>&<const>|<var2> 論理積

IOR:<var1>:<const>|<var2> <var1>←<var1>|<const>|<var2> 論理和

XOR:<var1>:<const>|<var2> <var1>←<var1>^<const>|<var2> 排他的論理和

RND:<var1>:<from>:<to> <var1>←範囲 <from>～<to> の一様乱数値

《同期》

WAIT 直前に実行した動作の終了（再生の終了）を待つ。

WAIT:<ms> 引数にしたミリ秒だけ待つ。<ms> = 1～30000。

（注：分解能は 32[ms]）

WAIT:SOUND 音再生の終了を待つ（単独で音再生中の判定にも使える）。

WAIT:LIGHT 光再生の終了を待つ（単独で光再生中の判定にも使える）。

《関数》

CALL:<label> <label> のサブルーチンをコールする。

※ネストは最大16個。パラメータ渡しは変数で代用することとする。

RETURN サブルーチンからのリターン。

※スタック・ポインタは無チェックなので、ネストの整合性に注意する必要がある。

【0162】RCODEのコマンドのうち、ロボット1の「動き」を制御するコマンドは以下の5つの基本形に大別される。すなわち、

【0163】（1）POSE ある姿勢（モーションの開始姿勢）をとらせる。

（2）MOVE あるモーションをさせる。OMNE/OMTE/OMLE 2_S/OMSE に対応。

（3）PLAY あるモーションを再生させる。OAllReplay/OHeadReplay/OLegsReplay/OTailReplay に相当する。

（4）STOP 動作の停止（現在行っているモーションを終了してから）。

（5）QUIT 動作の緊急停止（直ちに）。

SET:ROBOT:POWER:1

SET:ROBOT:POWER:0

GET:ROBOT

GET:ROBOT:<var>

SET:ROBOT:<var>:<value>

【0164】MOVEとPLAYは、組み込み動作を行うか、又は、メモリ・スティック上の ".mtn" ファイルに定義されている動作を行うかの違いだけで、「ある動作をさせる」という点では同じ機能であると言える。MOVE の動作は組み込みなので、カスタマイズできないが、細かいパラメータ指定等を行うことができる。これに対し、PL 40 AYの動作は "*.mtn" ファイルの再生なので、パラメータ指定はできないが、差し替え可能である。

【0165】以下に、RCODE コマンドの一覧を示しておく。

【0166】

* 【数2】

電源ON

電源OFF

全変数ダンプ

変数 取得

変数 設定

POSE:ROBOT:<motion_name>	motionの開始ポーズへ
POSE:LEGS:<motion_name>	motionの開始ポーズへ
POSE:HEAD:<motion_name>	motionの開始ポーズへ
POSE:TAIL:<motion_name>	motionの開始ポーズへ
PLAY:ROBOT:<motion_name>	motion再生
PLAY:LEGS:<motion_name>	motion再生
PLAY:HEAD:<motion_name>	motion再生
PLAY:TAIL:<motion_name>	motion再生
PLAY:SOUND:<sound_name>:<volume>	音再生
volume = 0~100	
PLAY:LIGHT:<pattern_name>:<times>	光再生 (LEDパターン)
times = 0~16 (0:Default 16:Loop)	
MOVE:ROBOT:<motion_name>	OMLE組み込み動作再生 (転倒復帰など)
MOVE:LEGS:WALK:<style1>:<dir>:<times>	歩行
MOVE:LEGS:STEP:<style1>:<dir>:<times>	歩行 (StepWalk)
MOVE:LEGS:KICK:<style2>:<deg>	蹴り
style1 = 0~10,12(R-Turn),13(L-Turn)	
style2 = 14(R-Kick),15(L-Kick)	
dir = 1~6	
times = 0~9999	
deg = -90~90	
MOVE:HEAD:HOME	頭部ホーム・ポジション
MOVE:HEAD:ABS:<tilt>:<pan>:<roll>:<time>	頭部 絶対位置移動
MOVE:HEAD:REL:<tilt>:<pan>:<roll>:<time>	頭部 相対位置移動
MOVE:HEAD:C-TRACKING	頭部カラー・トラッキング
MOVE:HEAD:C-TRACKING:<time>	頭部カラー・トラッキング
tilt = -180~180	
pan = -180~180	
roll = -180~180	
time = 0~99999 [ms]	
MOVE:TAIL:HOME	尾部ホーム・ポジション
MOVE:TAIL:ABS:<tilt>:<pan>:<time>	尾部絶対位置移動
MOVE:TAIL:SWING:<tilt>:<pan>:<time>	尾部スイング
tilt = -90~90	
pan = -90~90	
time = 0~99999 [ms]	
STOP:ROBOT	全体通常停止
STOP:LEGS	四肢通常停止
STOP:HEAD	頭部通常停止
STOP:TAIL	尾部通常停止
STOP:SOUND	音再生停止
STOP:LIGHT	光再生停止
QUIT:ROBOT	全体緊急停止
QUIT:LEGS	四肢緊急停止
QUIT:HEAD	頭部緊急停止
QUIT:TAIL	尾部緊急停止
QUIT:SOUND (STOP:SOUNDと等価)	音再生停止
QUIT:LIGHT (STOP:LIGHTと等価)	光再生停止

【0167】また、RCODEは、ロボット1の各部に 50 対して動作・制御値を指示したり、各部の状態を取得する (センサ出力値の読み出しなど) ために、システム変数を定義している。以下に、RCODEのシステム変数

の一覧を示しておく。

*【数3】

【0168】

*

Power	電源	0:OFF 1:ON
Head_tilt	頭部第1関節角度[°]	
Head_pan	頭部第2関節角度[°]	
Head_roll	頭部第3関節角度[°]	
Head_mouth	頭部顎関節角度[°]	
Tail_1	尾部第1関節角度[°]	
Tail_2	尾部第2関節角度[°]	
Leq_RF_1	右前脚第1関節角度[°]	
Leq_RF_2	右前脚第2関節角度[°]	
Leq_RF_3	右前脚第3関節角度[°]	
Leq_LF_1	左前脚第1関節角度[°]	
Leq_LF_2	左前脚第2関節角度[°]	
Leq_LF_3	左前脚第3関節角度[°]	
Leq_RR_1	右後脚第1関節角度[°]	
Leq_RR_2	右後脚第2関節角度[°]	
Leq_RR_3	右後脚第3関節角度[°]	
Leq_LR_1	左後脚第1関節角度[°]	
Leq_LR_2	左後脚第2関節角度[°]	
Leq_LR_3	左後脚第3関節角度[°]	
Head_sw	頭部圧力センサ	[10~3Pa]
Distance	頭部障害物センサ	[mm]
Leq_RF_sw	右前脚肉球センサ	[On:-1 Off:0]
Leq_LF_sw	左前脚肉球センサ	[On:-1 Off:0]
Leq_RR_sw	右後脚肉球センサ	[On:-1 Off:0]
Leq_LR_sw	左後脚肉球センサ	[On:-1 Off:0]
Gsensor_status	Gセンサ・ステータス	16ビット・フラグ
Gsensor_roll	Gセンサ・ロール角[°]	
Gsensor_pitch	Gセンサ・ピッチ角[°]	
Gsensor_yaw	Gセンサ・ヨー角[°]	
Cdt_npixel	色センサ画素数	[Pixels]
Psd_status	障害物センサ状況	
Psd_range	障害物までの距離	
Touch_head	頭部タッチ・センサ	
Touch_head_time	頭部タッチ・センサ押下時間	
Touch_head_press	頭部タッチ・センサ平均圧力	[単位不明]
Touch_RF	右前脚肉球センサ	ON→OFF/OFF→ON検出
Touch_LF	左前脚肉球センサ	ON→OFF/OFF→ON検出
Touch_RR	右後脚肉球センサ	ON→OFF/OFF→ON検出
Touch_LR	左後脚肉球センサ	ON→OFF/OFF→ON検出
Tone_num	トーン検出	音番号
Tone_level	音の大きさ	
Tone_dir	検出方向	
Melody_id	メロディ検出	メロディID
Melody_num	トーン数	(1~3)
Tone1_num	Tone1音番号	
Tone1_level	Tone1音の大きさ	
Tone1_dir	Tone1検出方向	

31

32

Tone2_num	Tone2音番号
Tone2_level	Tone2音の大きさ
Tone2_dir	Tone2検出方向
Tone3_num	Tone3音番号
Tone3_level	Tone3音の大きさ
Tone3_dir	Tone3検出方向
Sound_status	サウンド検出 ステータス
Sound_num	音番号
Sound_level	音の大きさ
Sound_dir	検出方向
Sound_busy	1: サウンド再生中 0: サウンド無し
Light_busy	1: Light再生中 0: Light無し

【0169】上記のシステム変数のうちGsensor_status * 【0170】
は16ビットのフラグで構成されるが、各ビットの意味 【表1】
は以下の通りである。 *

ビット	OMGsensor 状態	説明
0x0001	FALL_DOWN_FRONT	(前方) 転倒
0x0001	FALL_DOWN_RIGHT	(右側) 転倒
0x0001	FALL_DOWN_LEFT	(左側) 転倒
0x0001	FALL_DOWN_REAR	(後方) 転倒
0x0002	HOLD_UP	抱き上げ
0x0004	HOLD_DOWN	抱き下げ
0x0400	GET_UP	転倒復帰
0x0800	JOINT_DANGER	関節挟み込み
0x1000	JOINT_GAIN_ENABLED	ゲインON
0x2000	JOINT_GAIN_DISABLED	ゲインOFF

【0171】また、システム変数Sound_statusは、以下 ※ 【0172】
のような値によってロボット1への音声入力状態を表す 【表2】
ようになっている。 ※

Sound_status	状態	説明
0	ontonePEAK	ピークのある音
1	ontoneNOPEAK	ピークのない音
2	ontoneNOPOWER	音量が小さい
3	ontoneCALC	計算中
4	ontoneOVERFLOW	オーバーフロー

【0173】また、RCODEは、ロボット1上の各部位に配備されたセンサの出力データのログをとる（メモリ上の所定領域に一時記憶する）サービスを提供する。すなわち、RCODEコマンド"LOG:START"でロギング 40
の開始を指示するとともに、RCODEコマンド"LOG:END"でロギングの終了を指示することができる。また、変数"Log_n"によって記憶したログ件数を表す。

【0174】さらに、RCODEコマンド"LOG:PCFILE: <file>"によって、指定したファイル名でホスト・システム100上のファイルにログを保存することができる。同様に、RCODEコマンド"LOG:MSFILE: <file>"によって、指定したファイル名でメモリ・スティック上のファイルにログを保存することができる。

【0175】また、各センサの出力データを、メモリやファイルに保存することなく、逐次、プリンタで直接印刷出力するようにしてもよい。この場合、RCODEコマンド"LOG:PRINT: <s>:<t>:<i>"で出力形式を指定することができる。但し、sはセンサ番号を（【表3】を参照のこと）、tは値の種類を（【表4】を参照のこと）、iはインデックス（0～Log_n-1）を、それぞれ示すものとする。また、RCODEコマンド"LOG:PRINT:CRLF"で改行を、RCODEコマンド"LOG:PRINT:EOF"でファイル・クローズを、それぞれ指示することができる。

【0176】

【表3】

s	センサ	説明
0	Head Tilt	PRM:/r0/c0-Joint:j0
1	Head Pan	PRM:/r0/c0/c1-Joint:j1
2	Head Roll	PRM:/r0/c0/c1/c2-Joint:j2
3	Mouth	PRM:/r0/c0/c1/c2/c3-Joint:j3
4	Tail 1	PRM:/r1/c0-Joint:j0
5	Tail 2	PRM:/r1/c1-Joint:j1
6	Leg FR 1	PRM:/r2/c0-Joint:j0
7	Leg FR 2	PRM:/r2/c0/c1-Joint:j1
8	Leg FR 3	PRM:/r2/c0/c1/c2-Joint:j2
9	Leg FL 1	PRM:/r3/c0-Joint:j0
10	Leg FL 2	PRM:/r3/c0/c1-Joint:j1
11	Leg FL 3	PRM:/r3/c0/c1/c2-Joint:j2
12	Leg RR 1	PRM:/r4/c0-Joint:j0
13	Leg RR 2	PRM:/r4/c0/c1-Joint:j1
14	Leg RR 3	PRM:/r4/c0/c1/c2-Joint:j2
15	Leg RL 1	PRM:/r5/c0-Joint:j0
16	Leg RL 2	PRM:/r5/c0/c1-Joint:j1
17	Leg RL 3	PRM:/r5/c0/c1/c2-Joint:j2
18	Head Press	PRM:/r0/c0/c1/c2/c3-Sensor:t3
19	Distance	PRM:/r0/c0/c1/c2/p3-Sensor:p3
20	Acc 1	PRM:/acc0-Sensor:a0
21	Acc 2	PRM:/acc1-Sensor:a1
22	Acc 3	PRM:/acc2-Sensor:a2
23	Gyro 1	PRM:/gyro0-Sensor:g0
24	Gyro 2	PRM:/gyro1-Sensor:g1
25	Gyro 3	PRM:/gyro2-Sensor:g2
26	Thermo 1	PRM:/thermo0-Sensor:th0
27	Thermo 2	PRM:/thermo1-Sensor:th1
28	Button FR	PRM:/r2/c0/c1/c2/c3-Sensor:sw3
29	Button FL	PRM:/r3/c0/c1/c2/c3-Sensor:sw3
30	Button RR	PRM:/r4/c0/c1/c2/c3-Sensor:sw3
31	Button RL	PRM:/r5/c0/c1/c2/c3-Sensor:sw3

【0177】

* * 【表4】

t	定数名	実際のメンバー
0	VALUE	OSensorValue.value
1	SIGNAL	OSensorValue.signal
2	PWM	OJointValue.pwmDuty

【0178】既に説明したように、本実施例に係るオーサリング・ツールを用いてロボット1の動作制御用のシナリオを作成・編集すると、編集したシナリオを実現するためのRCODEに自動変換するようになっている。例えば、図13に示すようなGUI画面上で生成されたメイン・ルーチンを持つサッカー・ゲーム用シナリオの動作制御プログラムは、以下に示すようなRCODEを用いて以下のように記述される。

【0179】

【数4】

SETV:ROBOT:POWER:1

SET:Trace:1

SET:mode:0 // 0:サーチモード 1:追っ掛けモード

SET:head:0 // サーチモードで頭をグルグルするためのカウン

SET:lost:0 // ボールを見失った回数

:100 // MainLoop

SET:stat:Gsensor_status

AND:stat:1

IF:=:stat:1:9000 //ずっこけた?

IF:=:mode:0:1000 //サーチモードへ

IF:=:mode:1:2000 //追っ掛けモードへ

40 GO:100

:1000 // mode:0 サーチモード

SET:mode:0

MOVE:LEGS:STEP:RIGHT_TURN:0:4 //右回りしながらボールを捜す

SWITCH:head //頭をグルグルまわす処理

CASE:0:MOVE:HEAD:ABS:-15:0:0:500

CASE:1:MOVE:HEAD:ABS:-15:-40:0:500

CASE:2:MOVE:HEAD:ABS:-15:-80:0:500

CASE:3:MOVE:HEAD:ABS:-45:-40:0:500

50 CASE:4:MOVE:HEAD:ABS:-45:0:0:500

35

```

CASE:5:MOVE:HEAD:ABS:-45:40:0:500
CASE:6:MOVE:HEAD:ABS:-45:80:0:500
CASE:7:MOVE:HEAD:ABS:-15:40:0:500
CASE:8:MOVE:HEAD:ABS:-15:0:0:500
ADD:head:1 // head の インクリメント
MOD:head:9
WAIT
IF:<:Cdt_npixel:32:100 // ボール見えていれば 追っ掛けモードへ
:2000 // 追っ掛けモード
SET:mode:1
IF:<:Cdt_npixel:32:1000 // ボール見えているか?
MOVE:HEAD:C-TRACKING:100 // 色追跡開始
IF:>:Head_tilt:-58:2300 // 頭の角度で距離推定
IF:>:Head_pan:0:2210:2220 // 近ければキック!
:2210
MOVE:HEAD:HOME // 自分のアゴを蹴らないように (^^;)
  ちょと不自然
MOVE:LEGS:KICK:LEFT_KICK:0
MOVE:LEGS:STEP:SLOW:FORWARD:1
GO:2900
:2220
MOVE:HEAD:HOME // 自分のアゴを蹴らないように (^^;)
  ちょと不自然
MOVE:LEGS:KICK:RIGHT_KICK:0
MOVE:LEGS:STEP:SLOW:FORWARD:1
GO:2900
:2300 // 頭の角度により、ボールに近づく
CSET:>:Head_pan:60:1
CSET:>:Head_pan:45:2

```

```

//
// スクリプト形式で記述されたサッカー・ゲーム
//
#define ROBOT_NEUTRAL (0) // ニュートラル
#define ROBOT_FOUND (1) // ボール見つかった
#define ROBOT_LOST (2) // ボール見つからない
#define ROBOT_FALL (99) // ROBOT転倒
// ピンク
#define PinkBall (32)
// システム変数
extern int Head_tilt, Head_roll;
extern int Head_pan;
extern int Cdt_npixel;
extern int Gsensor_status;
// ボールを見つけたか見失ったか
int bBall = 0, bBoil;
// 現在の状態
int nSoccer = 0;
int nLost = 0; // ボールを見失った時のステップ数
// 関数の前方宣言

```

36

```

* CSET:>:Head_pan:15:3
CSET:<:Head_pan:-60:4
CSET:<:Head_pan:-45:5
CSET:<:Head_pan:-15:6
CSET:=:0:0:0
CASE:0:MOVE:LEGS:STEP:SLOW:FORWARD:4
CASE:1:MOVE:LEGS:STEP:SLOW:FORWARD:4
CASE:2:MOVE:LEGS:STEP:LEFT_TURN:0:4
CASE:3:MOVE:LEGS:STEP:SLOW:LEFT:4
10 CASE:4:MOVE:LEGS:STEP:SLOW:LEFTFORWARD:4
CASE:5:MOVE:LEGS:STEP:RIGHT_TURN:0:4
CASE:6:MOVE:LEGS:STEP:SLOW:RIGHT:4
CASE:7:MOVE:LEGS:STEP:SLOW:RIGHTFORWARD:4
:2900
WAITGO:100
:9000 // 転倒回復
QUIT:ROBOT
MOVE:ROBOT:ReactiveGU
WAIT
20 GO:100
【0180】また、図5を参照しながら説明したように、本実施例に係るオーサリング・システムでは、例えばC言語のような高級言語を用いてスクリプト形式で動作制御プログラムを作成・編集することができる。上記の【数4】で示したRCODEプログラムと等価な機能及び動作手順を実現するスクリプトを以下に示しておく。但し、スクリプトはC言語の文法に則って記述されているものとする。
【0181】
*30 【数5】

```

37

38

```

int SoccerDog(int);
void NeutralROBOT();
void FindBall();
void SearchBall();
void RecoveryGU();
void NearBall();
void KickBall();
//      メイン
void main( void ){

    //      Soccerフラグ
    int bSoccerFlag;
    int dummy = 0, true = 1;

    bSoccerFlag = dummy = true;

    //      電源ON
    PowerOn();

    //      メインループ
    while( bSoccerFlag ){
        //      サッカードッグ
        bSoccerFlag = SoccerDog(1);
    }

    //      電源OFF
    PowerOff();

}

//      サッカー開始
int SoccerDog(int dummy ){

    //      現在の状態で処理を分ける
    switch( nSoccer ){
    case 0: //ROBOT_NEUTRAL:      //      ニュートラル状態
        NeutralROBOT();
        break;
    case 1: //ROBOT_FOUND:      //      ボールを見つけた
        FindBall();
        break;
    case 2: //ROBOT_LOST:      //      ボールを見失った
        SearchBall();
        break;
    case 99: //ROBOT_FALL:      //      アイゴ転倒
        RecoveryGU();      //      転倒復帰処理
    }

    //      続行
    return 1; //FALSE;
}

```

39

40

```
// ボールを見つけ、近づく処理
void FindBall( void ){
    int nHead;
    nHead = Head_tilt;

    // カラートラッキング
    RcodeColorTracking(100);

    // 頭の角度でボールの距離を測定
    if( nHead > -58 ){
        // ボールを検出
        if (Cdt_npixel < 32){
            // ボールを見失った
            nSoccer = 2; //ROBOT_LOST;
        }
        else{
            // ボールに近づく
            NearBall();
        }
    }
    else{
        // ボールを蹴る
        KickBall();
    }
}

// ボールに近づく
void NearBall( void ){

    // 首の角度でボールの位置を設定する
    if( Head_pan == 0 ){
        RcodeStepWalk(0, 1, 4); //StepWalk(FORWARD, 4);
        RcodeWait();
        return;
    }
    if( Head_pan > 60 ){
        RcodeStepWalk(0, 1, 4); //      StepWalk(FORWARD, 4);
        RcodeWait();
        return;
    }
    if( Head_pan > 45 ){
        RcodeStepWalk(0, 13, 4); //      StepWalk(LEFT_TURN, 4);
        RcodeWait();
        return;
    }
    if( Head_pan > 15 ){
        RcodeStepWalk(0, 5, 4); //      StepWalk(LEFT, 4);
        RcodeWait();
        return;
    }
}
```

```
41
42
if( Head_pan < -60 ){
    RcodeStepWalk(0, 3, 4); //      StepWalk(LEFTFORWARD, 4);
    RcodeWait();
    return;
}
if( Head_pan < -45 ){
    RcodeStepWalk(0, 1, 4); //      StepWalk(RIGHT_TURN, 4);
    RcodeWait();
    return;
}
if( Head_pan < -15 ){
    RcodeStepWalk(0, 4, 4); //      StepWalk(RIGHT, 4);
    RcodeWait();
    return;
}

//      ボールを検出する
if (Cdt_npixel < 32){
    nSoccer = 2; //ROBOT_LOST:
    return;
}
}
//      ボールを蹴る
void KickBall( void ){

    //      頭の位置調整
    RcodeHead(0, 0, 0, 0);

    //      ボールが頭の左右どちら側にあるかで蹴る足を決める
    if( Head_pan > 0 ){
        RcodeKick(15, 0 );      //      左足
    }else{
        RcodeKick(14, 0 );      //      右足
    }

    //      歩行に移行するために1ステップ踏む
    RcodeStepWalk(0, 0, 1); //      StepWalk(FORWARD, 1);
}

//      ボールを探す処理
void SearchBall( void ){

    //      頭を動かす
    switch( nLost ){
    case 0:
        RcodeHead(-15, 0, 0, 500 );
        break;
    case 1:
        RcodeHead(-15, -40, 0, 500 );
        break;
    }
```


43

44

```

case 2:
    RcodeHead(-15, -80, 0, 500 );
    break;
case 3:
    RcodeHead(-45, -40, 0, 500 );
    break;
case 4:
    RcodeHead(-45, 0, 0, 500 );
    break;
case 5:
    RcodeHead(-45, 40, 0, 500 );
    break;
case 6:
    RcodeHead(-45, 80, 0, 500 );
    break;
case 7:
    RcodeHead(-15, 40, 0, 500 );
    break;
case 8:
    RcodeHead(-15, 0, 0, 500 );
    break;

}

// ステップ数の設定
nLost = (nLost + 1) % 9;

// 右側に歩く
RcodeStepWalk(12, 1, 4); //StepWalk(RIGHT_TURN, 4);

// 色検出
if (Cdt_npixel < 32){
    nSoccer = 2; //ROBOT_LOST;
}
else{
    // nLostの初期化
    nLost = 0;
    // ROBOTに近づく処理
    nSoccer = 1; //ROBOT_FOUND;
}
}

//
// ニュートラル時の処理
void NeutralROBOT( void ){

    // 転倒検出
    if (Gsensor_status & 1){
        // 転倒
        nSoccer = 99; //ROBOT_FALL;
        return;
    }

```

```

45      }

      //      転倒はしていない。ボール検出
      if (Cdt_npixel < 32){
          //      ボールを見失った
          nSoccer = 2; //ROBOT_LOST;
      }
      else{
          //      ボールを見つけた
          nSoccer = 1; //ROBOT_FOUND;
      }

    }
    //      転倒復帰
    void RecoveryGU( void ){

        //      転倒復帰成功?
        RcodePlay("ReactiveGU");
        RcodeWait();

        //      現在の状態をニュートラルにする
        nSoccer = 0; //ROBOT_NEUTRAL;
    }

```

【0182】本実施例に係るオーサリング・システムは、ユーザがテキスト・エディタなどを用いて作成・編集したスクリプトを、RCODE形式のプログラムに自動変換することができる（前述）。したがって、[数5]に示したスクリプトは、本実施例に係るオーサリング・ツールを用いて、[数4]に示したようなRCODE形式のプログラム・コードに自動変換される。

【0183】例えば、RCODEを用いて動作処理プログラムを編集するホスト・システム100と、動作処理プログラムを実行するロボット1とが、bluetoothや、11Bのような無線データ通信（あるいは、その他の無線又は有線の通信手段）を介して接続されている場合、ホスト・システム1は1行すなわち1ステップ単位でRCODEプログラムを送信し、ロボット1側ではインタープリタがこれを逐次解釈し実行することができる。また、コマンド転送時に、ホスト・システム1側は送信データを暗号化してもよい。

【0184】本実施例に係るオーサリング・ツールは、RCODE形式で記述されたスクリプトの編集作業、並びに、ロボット1側との協働的動作によるリアルタイムのデバッグ作業のために、編集ウィンドウを用意している。図6に示した編集ウィンドウと区別するために、この編集ウィンドウのことを、以下では、「スクリプト編集ウィンドウ」と呼ぶことにする。

【0185】図16には、スクリプト編集ウィンドウの構成を図解している。

【0186】スクリプト編集ウィンドウは、7個のコン

ボ・ボックスを備えており、階層メニュー形式でコマンドを指定することができる。同図に示す例では、編集領域の第1行目のコマンドを指定の様子が示されている。

【0187】コンボ・ボックスの下にある[Command]ボタンをクリックすると、指定したコマンドをロボット1側に転送するようになっている。転送方式は、bluetoothや、11Bなどの無線データ通信を基本とするが、その他の通信方式であってもよい。また、ここで行われるコマンド転送は、後述の[Execute]とは非同期に動作する。

【0188】また、[Command]ボタンの左横にある[↓]ボタンをクリックすると、指定したコマンド文字列が、下方の編集領域に貼り付けられる。

【0189】[Command]ボタンの下方に配設された編集領域上では、キー入力によりスクリプトを直接編集することができる。

【0190】また、編集領域のさらに下方には、[Open]、[Save]、[Send]、[Execute]、及び、[Stop]という各ボタンが配設されている。

【0191】[Open]ボタンのマウス操作により、ホスト・システム100のローカル・ディスクなどから所望のスクリプト・ファイルを選択して、編集領域に表示することができる。

【0192】[Save]ボタンのマウス操作により、編集領域に表示されているスクリプトを、ファイルに保

存することができる。

【0193】[Send] ボタンをクリックすることにより、編集領域に表示されているスクリプトをロボット1側に転送することができる。転送方式は、Bluetoothや、11Bなどの無線データ通信を基本とするが、その他の通信方式であってもよい。但し、スクリプトは、転送されるだけで、ロボット1上でそのまま実行される訳ではない。なお、[Send] 処理の終了時には、コンソール・ウィンドウ（図示しない）で“END”という行が表示される。

【0194】[Execute] ボタンをクリックすることによって、ロボット1側に送信したスクリプトの実行開始を指示することができる。

【0195】[Stop] ボタンをクリックすることによって、ロボット1において実行中のスクリプトの強制停止を指示することができる。

【0196】C. ロボットによる動作プログラムの実行／デバッグ

上述したように、スクリプト編集ウィンドウ上で[Command] ボタンをクリックすることによって、該当する1行のコマンドをロボット1側に転送することができる。また、[Send] ボタンを押すことによって、編集領域に開かれているスクリプトをロボット1側に転送することができる。このようなコマンド転送時には、コマンド・データを一旦暗号化してから送出することが好ましい。

【0197】他方のロボット1側では、RCODEなどで記述された動作制御プログラムの実行及びデバッグ環境として、インタープリタ／デバッガと、ミドルウェアと、ドライバと、オペレーティング・システム（OS）とを備えている（図5を参照のこと）。

【0198】インタープリタは、RCODE形式で記述されたプログラムを1行ずつ読み込んで解釈して実行する高水準言語プログラムである。また、デバッグ時などにおいて、ホスト・コンピュータ100側から暗号化された形式でRCODEプログラムが送信される場合には、インタープリタは、これを一旦復号化してから解釈・実行を行う。

【0199】デバッガは、RCODEプログラム中の誤り（バグ）を発見して、修正する作業を支援するプログラムである。すなわち、プログラムを指定した行で実行を止めたり、そのときのメモリや変数の内容を参照することができる。

【0200】ミドルウェアは、ロボット1における歩行その他の脚式動作などの制御や、カメラ15からの入力画像の認識処理や、マイク16からの音声入力の認識処理などを行う処理エンジンである。ミドルウェアは、ドライバを介して各関節アクチュエータやその他のハードウェアの操作を行うことができる。

【0201】ミドルウェアは、ロボット1の脚式動作制

御において、目標とされる姿勢や目標とされる動作に遷移するための情報（ここでは、仮に「姿勢遷移情報」と呼ぶことにする）を生成する。すなわち、インタープリタなど上位プログラムから供給される行動指令情報に基づいて、ロボット1の現在の姿勢又は動作から次の姿勢又は動作（目標とされる姿勢又は目標とされる動作）に遷移させるための姿勢遷移情報を生成する。現在の姿勢から次に遷移可能な姿勢は、胴体や手や脚の形状、重量、各部の結合状態のようなロボット1の物理的形狀や、各関節の曲がる方向や角度などの装置構成によって決定される。ミドルウェアは、このような装置構成を考慮して姿勢遷移情報を生成する。

【0202】本実施例では、ミドルウェアは、ロボット1が遷移可能な姿勢及び遷移する際の動作を、例えば有向グラフの形式であらかじめ登録しており、有向グラフに従って上位プログラムからの行動指令を姿勢遷移情報に変換して、ドライバに対する具体的な動作指示を発行するようになっている。

【0203】一般に、ロボット1は、現在の姿勢によっては、RCODEで記述された指令内容に従って直接遷移できない場合がある。すなわち、ロボット1の姿勢は、現在の姿勢から直接遷移可能な姿勢と、直接には遷移できなくてある動作や姿勢を経由してなら可能となる姿勢とに分類される。

【0204】例えば、ロボット1が図1に示すように4足歩行型のロボット装置である場合、手足を大きく投げ出して寝転んでいる状態から伏せた状態へ直接移動することはできるが、立った姿勢へ直接遷移することはできず、一旦手足を胴体近くに引き寄せて伏せた姿勢を経由して、それから立ち上がるという少なくとも2段階の動作が必要である。

【0205】また、このように2段階の動作を経由しなければ安全に（すなわちロボット1が転倒することなく）実行できないような姿勢遷移も存在する。例えば、図1に示すような4足歩行型のロボット1は、立っている姿勢で両前足を挙げて「万歳」をしようすると転倒してしまう場合がある。あるいは、現在の姿勢が寝転び姿勢（寝姿勢）にあるときに、指令の内容として座り姿勢しかできないような「足をバタバタさせる」といった内容が送られてきたとき、そのまま足をバタバタさせる命令を各関節駆動部に発行してしまうと、寝姿勢から座り姿勢への遷移と足をバタバタさせる動作とが実行されてしまうことになる。この結果、ロボット1はバランスを崩して転倒してしまうであろう。

【0206】本実施例では、ミドルウェアは、ロボット1がとり得る姿勢及び動作が登録されるとともに、各姿勢を遷移させる動作で結んだ有向グラフを保持している。そして、現在の姿勢から目標とされる（すなわち指示された）姿勢又は動作までの経路を、この有向グラフ上で探索して、その探索結果に基づいて、現在の姿勢か

ら目標とされる姿勢又は動作へと遷移させるような命令を生成するようになっている。

【0207】図17には、本実施例において使用される有向グラフの具体例を図解している。有向グラフは、ロボット1がとり得る姿勢を示すノードと、遷移可能な2つのノード（姿勢）間を結ぶ有向アーク（動作アーク）とで構成される。図17に示す例では、ノードすなわち姿勢として、「ねそべる」、「ふせる」、「すわる」、「たつ」、「あるく」という5種類が含まれる。また、有向アークは、ある姿勢から他の姿勢へ遷移する動作アークと、他の姿勢からある姿勢へ戻る動作アークと、1つのノード内で動作が完結する自己動作アークとがある。また、ある姿勢から他の姿勢へ遷移する動作アークが複数あることや、ある姿勢について複数の自己動作アークがあることもある。

【0208】例えば、ロボット1の現在の姿勢が「ふせる」であるときに、インタープリタに対して「すわる」ことを指示するRCODEコマンドが投入された場合、図17に示すように、「ふせる」と「すわる」の各ノード間には有向アークが存在するので直接遷移が可能である。したがって、ミドルウェアは、インタープリタから渡される指示に従って、ドライバに対して駆動指令を発行すればよい。

【0209】これに対し、現在の姿勢が「ふせる」であるときに、インタープリタに対して「あるく」ことを指示するRCODEコマンドが投入された場合、図17に示すように、「ふせる」というノードから「あるく」というノードへ直接結ぶ有向アークは存在しない。したがって、ミドルウェアは、有向グラフ上でノード「ふせる」からノード「あるく」へ向かう経路探索して、ノード「たつ」を経由してノード「ふせる」からノード「あるく」に遷移するという姿勢遷移計画を立て、該計画に基づいてドライバに対して駆動指令を発行する。

【0210】以上を略言すれば、本実施例によれば、ミドルウェアがロボット1の姿勢・動作遷移特性を管理しており、RCODEコマンド・レベルでの論理的な姿勢・動作指示とハードウェア上の姿勢・動作遷移特性との差異を吸収できるメカニズムを提供している。

【0211】したがって、一般ユーザは、オーサリング・ツールを用いてGUI画面上でロボット1の動作シナリオを編集したり、あるいはスクリプト形式で動作制御プログラムを記述するような場合に、現実のロボット1における姿勢・動作遷移特性のような詳細なハードウェア情報を気にしなくて済む。また、ミドルウェア・レベルにおいてRCODEコマンド・レベルでの論理的な姿勢・動作指示とハードウェア上の姿勢・動作遷移特性との差異を吸収するので、ロボットの姿勢・動作遷移特性に反していることはRCODEプログラムのデバッグの対象とはならない。

【0212】〔追補〕以上、特定の実施例を参照しながら

ら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。

【0213】本実施例では、イヌを模した4足歩行を行うベット型ロボットを例に挙げて本発明に係るオーサリング・システムについて詳解したが、本発明の要旨はこれに限定されない。例えば、ヒューマノイド・ロボットのような2足の脚式移動ロボットや、あるいは脚式以外の移動型ロボットに対しても、同様に本発明を適用することができることを充分理解されたい。

【0214】あるいは、ロボットのような物理的な機械装置ではなく、コンピュータ・グラフィックスにより生成されるキャラクタを用いたアニメーションの動作シーケンスの作成・編集のために、本発明に係るオーサリング・システムを適用することも可能である。

【0215】要するに、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参照すべきである。

【0216】

【発明の効果】以上詳記したように、本発明によれば、ロボットの所定の動作パターンを記述する一連のコマンド／データを作成することができる、優れたオーサリング・システム及び方法を提供することができる。

【0217】また、本発明によれば、ロボットの動作状態を規定する部品の集合を用いて動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することができる。

【0218】また、本発明によれば、各部品をコンピュータ・ディスプレイ上に配置して動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することができる。

【図面の簡単な説明】

【図1】本発明を実施に供される四肢による脚式歩行を行う移動ロボット1の外観構成を示した図である。

【図2】移動ロボット1の電気・制御系統の構成図を模式的に示した図である。

【図3】制御部20の構成をさらに詳細に示した図である。

【図4】ホスト・コンピュータ100のハードウェア構成例を模式的に示した図である。

【図5】本実施例に係るオーサリング・システムの全体構成を模式的に示した図である。

【図6】本実施例に係るオーサリング・ツールが提供する、ユーザがロボット1の動作シナリオを作成・編集するためのGUI画面の構成を示した図である。

【図7】メニュー項目「ファイル」に関するプルダウン・メニューを示した図である。

【図8】メニュー項目「グループ」に関するプルダウン・メニューを示した図である。

【図9】編集ウィンドウ上に動作ボックス[A]及び分岐ボックス[B]が各1個配置されている様子を示した図である。

【図10】図9に示す編集ウィンドウ上で、[START]ボックスと動作ボックス[A]、並びに、動作ボックス[A]と分岐ボックス[B]を接続した様子を示した図である。

【図11】動作ボックスの詳細を指定するためのダイアログを示した図である。

【図12】分岐ボックスの詳細を指定するためのダイアログを示した図である。

【図13】ロボット1がサッカー・ゲームを行うためのシナリオを、オーサリング・ツールのGUI画面上で作成した様子を示した図である。

【図14】図13に示すサッカー・ゲームのシナリオ中のグループ・ボックス[探索モード]を別の編集ウィンドウとして開いた様子を示した図である。

【図15】図13に示すサッカー・ゲームのシナリオ中のグループ・ボックス[追跡モード]を別の編集ウィンドウとして開いた様子を示した図である。

【図16】本実施例に係るオーサリング・ツールが用意するスクリプト編集ウィンドウの構成を示した図である。

【図17】ロボット1側のRCODEプログラム実行環境において保持される、ロボット1の姿勢及び動作の選移を規定する有向グラフを示した図である。

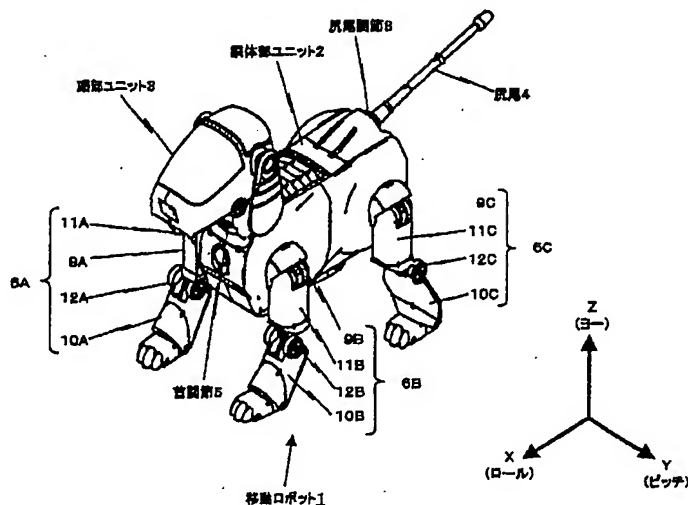
【符号の説明】

- 1…移動ロボット
2…胴体部ユニット

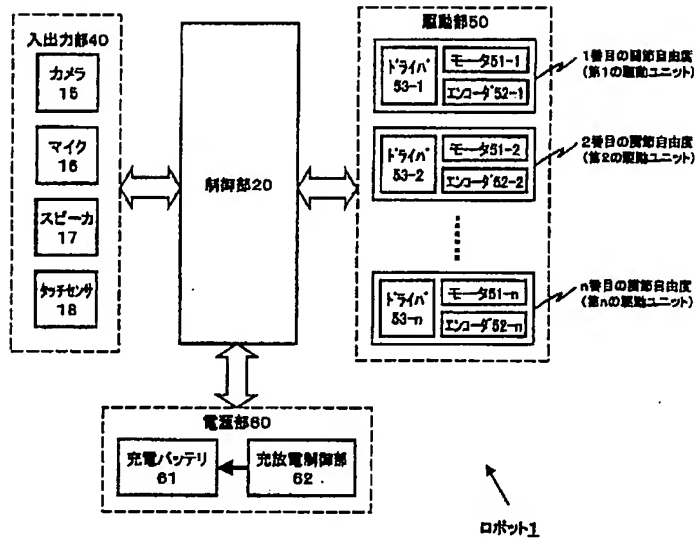
- * 3…頭部ユニット
4…尻尾
6A~6D…脚部ユニット
7…首関節
8…尻尾関節
9A~9D…大腿部ユニット
10A~10D…脛部ユニット
11A~11D…股関節
12A~12D…膝関節
15…CCDカメラ
16…マイクロフォン
17…スピーカ
18…タッチセンサ
20…制御部
21…CPU
22…RAM
23…ROM
24…不揮発メモリ
25…インターフェース
26…無線通信インターフェース
27…ネットワーク・インターフェース・カード
28…バス
29…キーボード
40…入出力部
50…駆動部
51…モータ
52…エンコーダ
53…ドライバ

*

【図1】

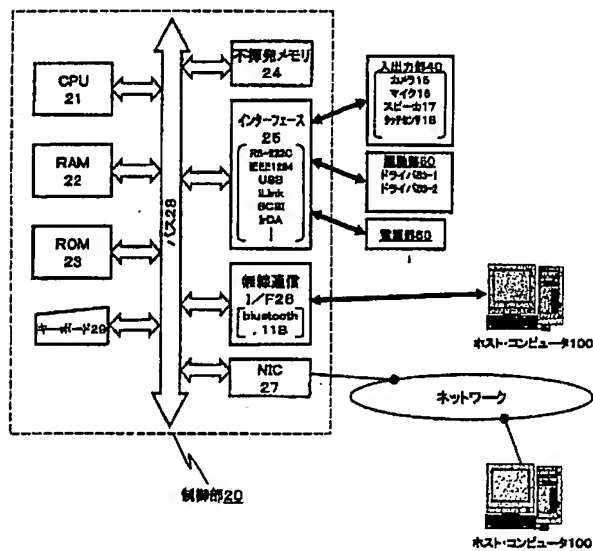


【図2】

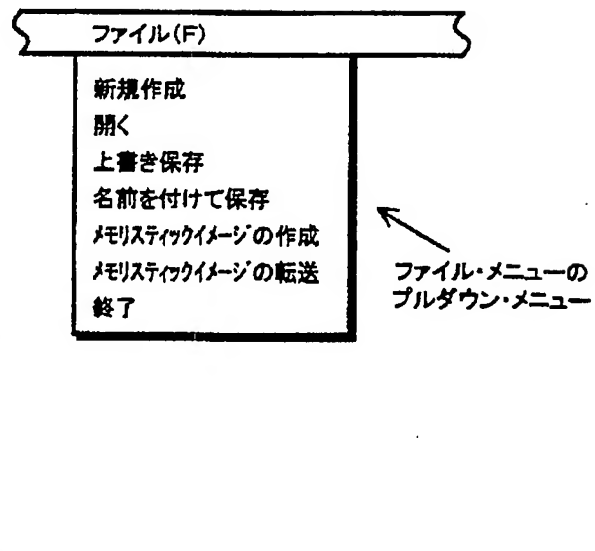
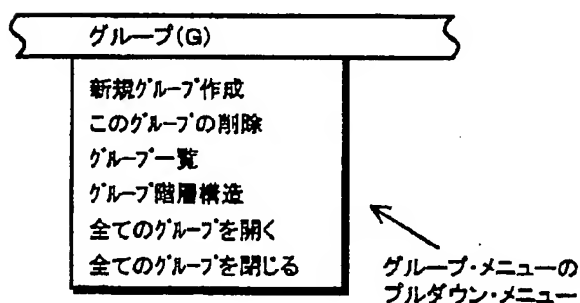


【図3】

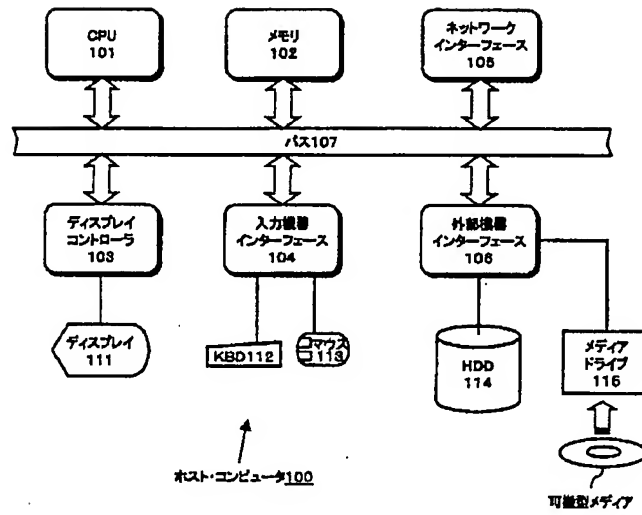
【図7】



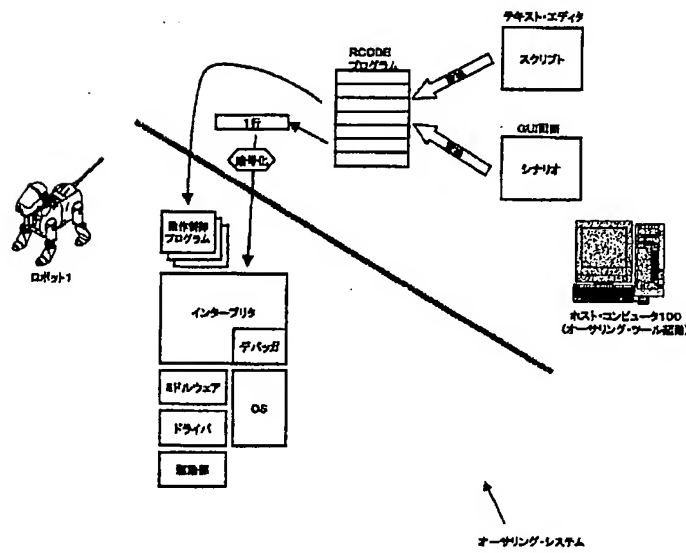
【図8】



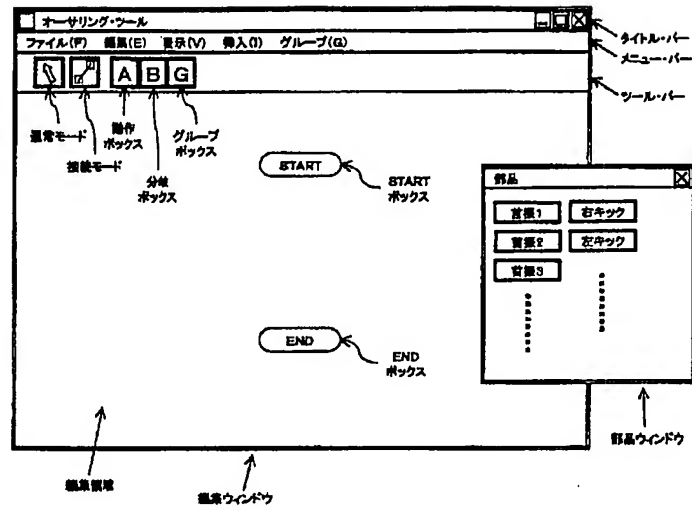
【図4】



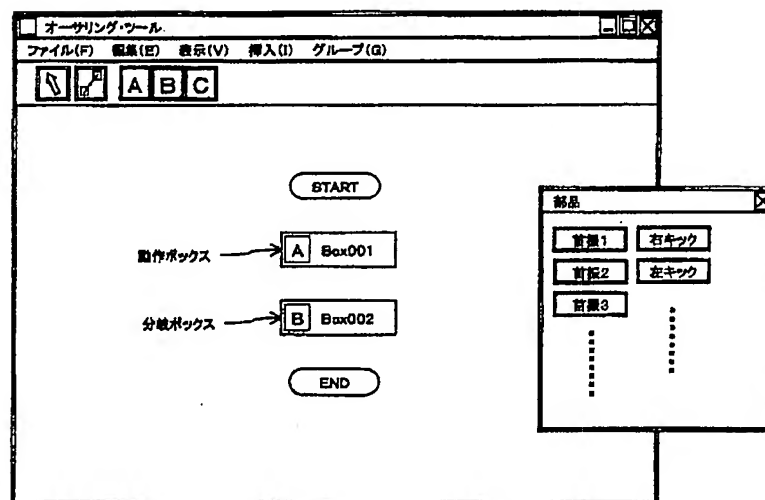
【図5】



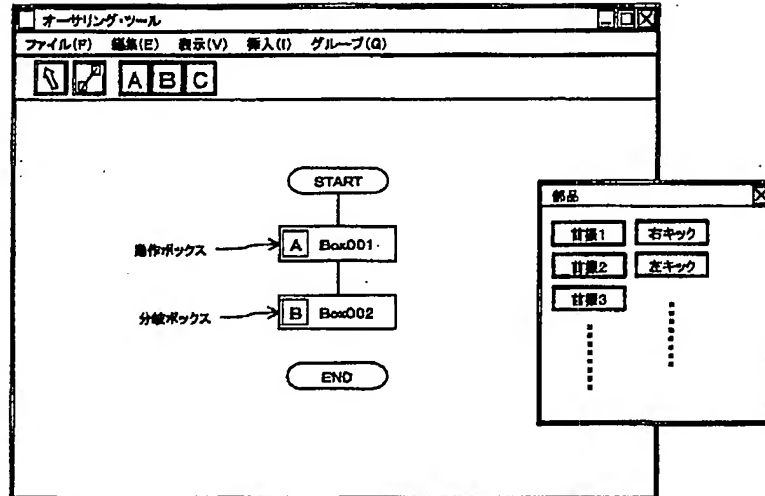
【図6】



【図9】



【図10】



【図11】

Action	Part	Name	arg1	arg2	arg3	arg4	Comment
SET		Power	1				
PLAY	ROBOT	Alubelp.D					
PLAY	SOUND	alp_000	100				

動作ボックスの閉鎖を
指示するためのダイアログ

【図12】

BranchBox

Name: モード分岐 Comment:

名前フィールド コメント・フィールド

Type Variable Op Value Comment

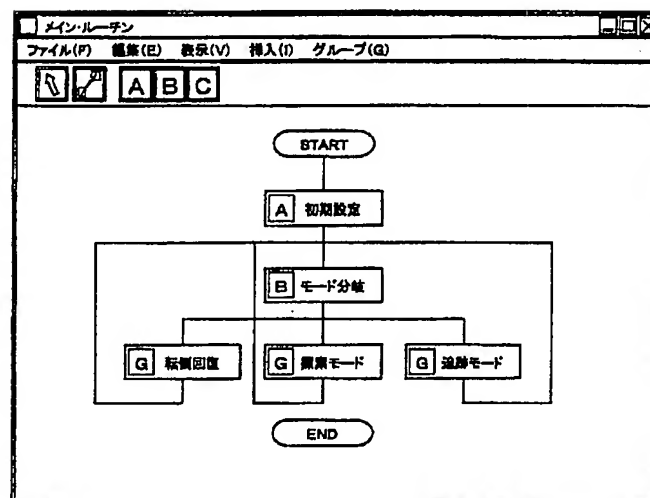
追加(A) 変更(M) 削除(D)

Type	Variable	Op	Value	JumpTo	Comment
IF	Gameover_status	Bit	1		強制終了
IF	mode	=	0		標準モード
IF	mode	=	1		過剰モード

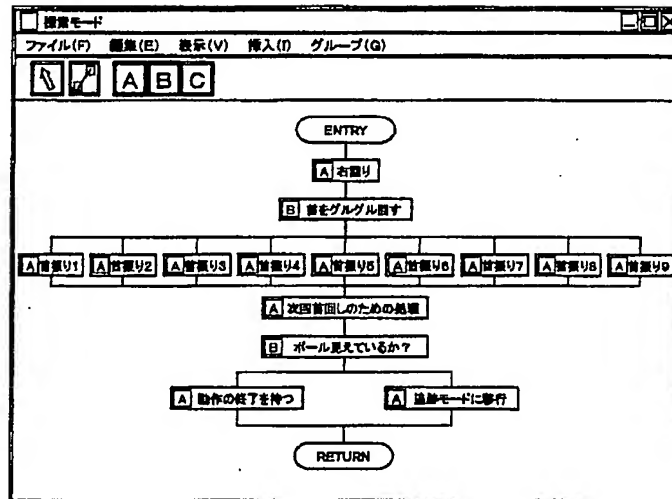
閉じる(C) コマンド・リスト

分岐ボックスの挙動を指定するためのダイアログ

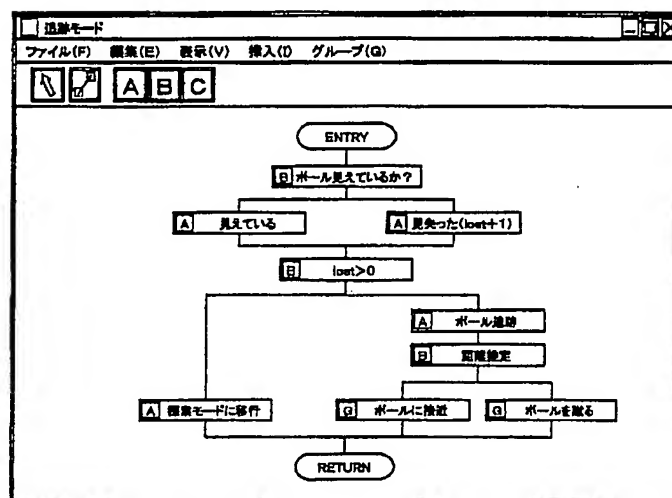
【図13】



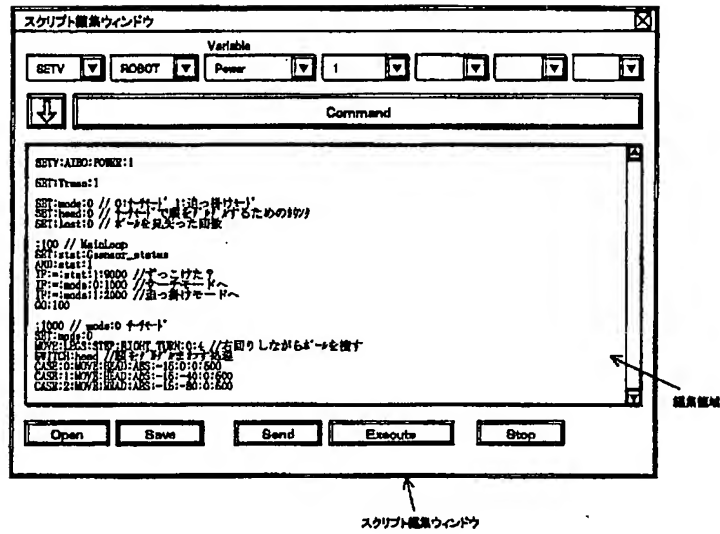
【図14】



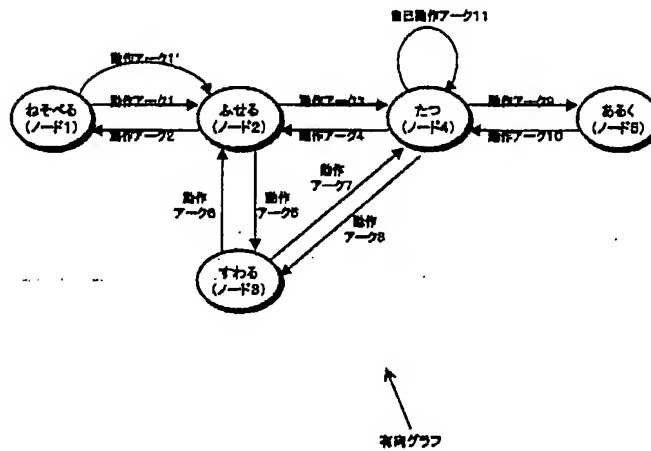
【図15】



【圖 16】



【圖 17】



フロントページの続き

F ターム (参考)

3F059	AA00	BA02	BB06	BC07	BC09
	CA05	CA06	DA02	DA05	DA08
	DB04	DB08	DB09	DC01	DC04
	DD01	DD04	DD08	DD11	DD18
	DE05	FA03	FA05	FB01	FB05
	FC02	FC07	FC13	FC14	
3F060	AA00	CA14	CA26	GA05	GA13
	GB25	GD13	GD15	HA02	HA32
	HA35				